UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSGRADO

PROGRAMA DE MAESTRÍA EN INGENIERIA INFORMATICA.

TÍTULO:

"ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE UNA APLICACIÓN WEB DESARROLLADA UTILIZANDO MARCOS DE TRABAJO DEL LADO SERVIDOR DJANGO Y LARAVEL."

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN INGENIERÍA INFORMÁTICA

ING. WALTER RICHARD PAIVA AYALA

EJECUTOR

PIURA PERÚ

(OCTUBRE – 2018)

DECLARACIÓN DE ORIGINALIDAD

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSGRADO

SECCIÓN DE POSGRADO EN INGENIERIA INFORMATICA

PROGRAMA DE MAESTRÍA EN

INGENIERÍA INFORMÁTICA

TESIS

"ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE UNA APLICACIÓN WEB DESARROLLADA UTILIZANDO MARCOS DE TRABAJO DEL LADO SERVIDOR DJANGO Y LARAVEL."

LOS SUSCRITOS DECLARAMOS QUE EL PRESENTE TRABAJO DE TESIS ES ORIGINAL, EN SU CONTENIDO Y FORMA

ING. WALTER RICHARD PAIVA AYALA
EJECUTOR

DR. MOISÉS DAVID SAAVEDRA ARANGO
ASESOR

UNIVERSIDAD NACIONAL DE PIURA

ESCUELA DE POSGRADO

SECCIÓN DE INGENIERIA INFORMATICA

PROGRAMA DE MAESTRÍA EN

INGENIERÍA INFORMÁTICA

"ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE UNA APLICACIÓN WEB DESARROLLADA UTILIZANDO MARCOS DE TRABAJO DEL LADO SERVIDOR DJANGO Y LARAVEL."

APROBADA EN CONTENIDO Y ESTILO POR:

DR. FLABIO ALFONSO GUTIERREZ SEGURA
PRESIDENTE

DR. VÍCTOR ANGEL ANCAJIMA MIÑÁN
SECRETARIO

DR. JONATHAN DAVID NIMA RAMOS
VOCAL

ESCUELA DE POSGRADO

UNIVERSIDAD NACIONAL DE PIURA

ACTA DE SUSTENTACIÓN

MAESTRÍA EN INGENIERÍA INFORMÁTICA

Los Miembros del Jurado Calificador que suscriben, reunidos para la sustentación de la Tesis, para optar el Grado Académico de Maestro en **INGENIERÍA INFORMÁTICA.** Presentada por:

PAIVA AYALA – WALTER RICHARD

Con el asesoramiento del DR. MOISES DAVID SAAVEDRA ARANGO denominada:

"ANÁLISIS COMPARATIVO DEL RENDIMIENTO DE UNA APLICACIÓN WEB DESARROLLADA UTILIZANDO MARCOS DE TRABAJO DEL LADO SERVIDOR LARAVEL Y DJANGO"

Oídas las respuestas y absueltas las observaciones formuladas, se declara:

APROBADO				DESAPROBADO
Excelente	*Sobresaliente*	*Bueno*	*Aceptable*	
	X			

En consecuencia, previa aprobación del Art.º 83, del Reglamento General de la Escuela de Posgrado, queda en condiciones de ser calificado **APTO** para obtener el Grado Académico de **MAESTRO EN INGENIERÍA INFORMÁTICA.** De conformidad con lo estipulado en la ley.

PIURA, MIERCOLES 14 DE AGOSTO DEL 2019.

DR. FLABIO ALFONSO GUTIERREZ SEGURA
PRESIDENTE

DR. VICTOR ANGEL ANCAJIMA MIÑAN
SECRETARIO

DR. JONATHAN DAVID NIMA RAMOS
VOCAL

Esquina Jr. Tacna / Jr. Apurimac, Apartado Postal 295 - Telf.: 073 - 328834 E-mail: posgrado@unp.edu.pe
http://www.unp.edu.pe/unp/postgrado - Piura - Perú
http://www.epg-unp.com/moodle/

DEDICATORIA

Dedico este trabajo principalmente a Dios por protegerme durante todo mi camino y darme fuerzas suficientes para superar obstáculos y dificultades, A mis padres, por ser el pilar más importante dentro de mis estudios que se concretan, sin importar los contrastes de opiniones, a mis hermanos, por su apoyo incondicional y por compartir conmigo buenos y malos momentos durante mis estudios, dándome la razón de llegar a una meta más, a mi mamita Fausta allá en el cielo, que con el ejemplo me enseñó a perseverar y a mi papá Juan por sus consejos y con esta frase que siempre recuerdo "todo lo que empieces termínalo", A mi hermanito Alfredo que desde el cielo intercede como un angelito para seguir adelante.

AGRADECIMIENTOS

Agradezco a Dios, a mis padres, mamá Lidia, mis hermanos y a mis profesores de la Facultad de Ingeniería Industrial, departamento de Informática de la Universidad Nacional de Piura que siempre estuvieron dándome el apoyo y por enseñarme a no rendirme ante nada y siempre perseverar a través de sus nobles consejos.

A mis padrinos y tíos por el apoyo moral que desde siempre estaban pendientes en mi desarrollo académico.

A mi amigo Bernardo y hermanos, por darme la oportunidad de aplicar mis conocimientos profesionales.

RESUMEN

En la presente investigación se planteó una comparación del rendimiento (tiempo de respuesta y uso de memoria RAM) de los marcos de trabajo (*frameworks*) del lado servidor Laravel y Django, aplicados al proceso de trámite documentario. Usando los factores: framework, número de usuarios y tipo de operación, en un diseño factorial 2x3x2. Los datos fueron obtenidos a partir del software Apache JMeter (tiempo de ejecución) y JConsole (memoria RAM). Tras procesarse estadísticamente se obtuvo que el rendimiento, se ve influenciado de manera significativa por los factores *framework* y t*ipo de operación*. Además, las medias marginales arrojan un tiempo de ejecución para Django (75.375 ms) menor al de Laravel (161.208 ms) y en uso de memoria RAM, Laravel (220,844 MB) es menor respecto al de Django (284.121 MB).

Palabras clave: marco de trabajo, *framework*, tiempo de respuesta, uso de memoria RAM, rendimiento, Laravel, Django.

ABSTRACT

In the present investigation a comparison of the performance (response time and use of RAM memory) of the frameworks of the server side Laravel and Django, applied to the process of documentary processing, was proposed. Using the factors: framework, number of users and type of operation, in a 2x3x2 factorial design. The data was obtained from the software Apache JMeter (runtime) and JConsole (RAM). After processing statistically it was obtained that the performance is significantly influenced by the framework and type of operation factors. In addition, the marginal means yield a time of execution for Django (75,375 ms) lower than that of Laravel (161,208 ms) and in RAM memory use, Laravel (220,844 MB) is lower than Django's (284.121 MB).

Keywords: framework, response time, RAM memory usage, performance, Laravel, Django

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

Toda empresa de desarrollo de software aspira a desarrollar aplicaciones de calidad y a su vez en el menor tiempo posible. Para poder hacerlo, actualmente se opta por usar herramientas denominadas frameworks o marcos de trabajo que permiten cumplir con esos objetivos. Sin embargo, la diversidad de frameworks hace que innegablemente cualquiera se cuestione cuál usar, o cuál es el mejor.

En este marco los framework PHP son muy populares en la actualidad por su sintaxis y simplicidad, criterios que también cumple el framework Django. Conforme aparecen nuevos frameworks y nuevas versiones de los ya existentes, también lo hacen foros, páginas, artículos respecto a que framework utilizar para poder garantizar el correcto funcionamiento de las aplicaciones a desarrollar.

Entre los puntos de comparación que son mencionados con mayor frecuencia están: arquitectura, curva de aprendizaje, seguridad, rendimiento, escalabilidad.

El presente trabajo de investigación tiene por objetivo analizar comparativamente el rendimiento de las aplicaciones desarrolladas con ambos marcos de trabajos y verificar si existe un efecto significativo entre las variables dependientes tiempo de respuesta y uso de memoria RAM (variables con las cuales se abarcar el rendimiento) usando los factores: framework, número de usuarios y tipo de operación, con los cuales se forma un diseño factorial 2x3x2. Se tomó como referencia el rendimiento ya que este abarca dos aspectos importantes: tiempo de respuesta y uso de memoria RAM.

Everts (2015), en uno de sus artículos Web, resume los resultados obtenidos de un estudio realizado en el año 2010 por TRAC Research a más de 300 empresas

internacionales, en el cual se determinó que 4.4 segundos era el retraso en los tiempos de respuesta a partir de los cuales una empresa empezaba a registrar pérdidas en sus ingresos por un monto promedio de $4,100 por hora.

Por otro lado, en un artículo publicado en Web Forefront (2019) se menciona que debido a que un disco duro es un medio más lento para obtener acceso que la RAM, un sistema operativo por diseño coloca los datos en la RAM para mejorar las velocidades de acceso y, por lo tanto, el rendimiento. En un momento dado, decenas o cientos de aplicaciones y sus datos compiten por una pieza de RAM en un sistema.

El análisis estadístico fue llevado a cabo utilizando la prueba ANOVA, luego de procesar los datos obtenidos a partir del software Apache JMeter y JConsole, con los cuales se pudo tomar las muestras correspondientes al tiempo de ejecución y uso de memoria RAM respectivamente.

Los resultados obtenidos permiten interpretar que existe un efecto significativo entre los factores mencionados y las variables dependientes. De manera más específica se pudo interpretar que Laravel posee un tiempo de ejecución estadísticamente mayor respecto a Django, debido a los valores obtenidos de las medias marginales.

Asimismo, se recomienda utilizar otros rangos para los usuarios concurrentes pues se generó algunos inconvenientes con los seleccionados. De igual manera para las unidades de medida a utilizar en las pruebas.

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. Descripción Problemática

Las empresas de desarrollo hoy en día tienen agendas muy sobrecargadas con una gran cantidad de clientes que desean automatizar sus tareas. Por ello recurren a lenguajes de programación y entornos, los cuales les proporcionen las herramientas para poder cumplir con las demandas del mercado laboral.

Actualmente una de las herramientas para desarrollo más utilizadas en las empresas son los marcos de trabajo, que básicamente son programas que permiten desarrollar sistemas en tiempos menores.

Sin embargo, la diversidad de frameworks hace que innegablemente cualquiera se cuestione cuál usar, o cuál es el mejor. Además, por cada framework se realizan cambios en sus versiones, tanto así, que en algunas ocasiones se vuelve irreconocible, por un posible cambio de paradigma de programación.

En este marco los framework PHP son muy populares en la actualidad por su sintaxis y simplicidad, criterios que también cumple el framework Django.

Una característica a tener en cuenta en los desarrollos de software es el rendimiento de las aplicaciones desarrolladas. Por rendimiento se entenderá al tiempo de ejecución de la aplicación desarrollada, así como al uso de memoria RAM por parte de la misma.

Con respecto al tiempo de ejecución Soasta (2016), evaluó el tiempo promedio de espera de los usuarios a las páginas Web, indicando que el 2014 se fue de 3.8 segundos, el 2015 de 2.4 segundos y que se esperaba al

2016 una proyección de 2 segundos, por lo cual hace imprescindible encontrar aquel marco de trabajo que produzca software dentro de límites indicados por los usuarios (Ver Anexo 1).

Otra arista del rendimiento es el uso de la memoria RAM, que es un recurso importante de la computadora debido a que, como afirma DELL (2018), en su página oficial, “mientras más RAM tenga el equipo, más grande será la encimera digital en la que se debe trabajar y más rápidamente se ejecutarán los programas”.

Así mismo, como menciona Rahm (2017), en la actualidad los navegadores Web pueden manejar múltiples procesos permitiendo trabajar una mayor cantidad de ventanas, sin embargo, es necesario encausar el uso de la memoria RAM, por ello realiza pruebas de uso de memoria en los navegadores (*browser memory usage*). (Ver Anexo 2)

1.2. Formulación del Problema

Como se mencionó en líneas anteriores, los usuarios necesitan aplicaciones Web que contemplen algunos requerimientos mínimos de rendimiento, visto este a través del tiempo de ejecución y uso de memoria RAM, por lo tanto, se formula la siguiente pregunta de investigación:

¿Cuál es el resultado del análisis comparativo del rendimiento de una aplicación desarrollada utilizando los marcos de trabajo Laravel y Django?

1.3. Objetivos de la Investigación

1.3.1. Objetivo General

Realizar el análisis comparativo del rendimiento de las aplicaciones desarrolladas con los marcos de trabajo de lado servidor Django y Laravel.

1.3.2. Objetivos Específicos

1. Comparar teóricamente los marcos de trabajo del lado del servidor Django y Laravel.
2. Determinar si existe variación respecto al tiempo de ejecución de las aplicaciones desarrolladas utilizando los frameworks Laravel y Django.
3. Determinar si existe variación respecto al uso de memoria RAM de las aplicaciones desarrolladas utilizando los frameworks Laravel y Django.

1.4. Hipótesis y operacionalización de variables

1.4.1. Hipótesis General

Existe variación en el rendimiento de las aplicaciones desarrolladas en Laravel y Django.

1.4.2. Hipótesis Específicas

HE1: Existe variación en el tiempo de ejecución de las aplicaciones desarrolladas en Laravel y Django.

HE2: Existe variación en el uso de memoria RAM de las aplicaciones desarrolladas en Laravel y Django.

1.4.3. Operacionalización de variables

Tabla 1. Operacionalización de variables

ÍTEM	VARIABLE	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERATIVA	UNIDAD DE MEDIDA	INSTRUMENTO DE MEDICIÓN
1	**V. Dependiente:** **Rendimiento de la aplicación**	Proporción entre el producto o el resultado obtenido y los medios utilizados.	Funcionamiento adecuado de una aplicación basado en: **-Tiempo promedio de ejecución** que tarda desde que se realiza la petición hasta que devuelve los resultados en la página Web. **Uso de Memoria RAM** promedio que se usa desde que se realiza la petición hasta que devuelve los resultados en la página Web.	Milisegundos MegaBytes(MB)	Software para medir el tiempo y memoria RAM. Guía de observación.

3	**V. Independiente: Factores de la comparativa**	Elementos que influyen sobre el rendimiento de la aplicación	Los elementos definidos fueron los siguientes: - **Nº de usuarios:** Individuo(s) que utiliza o trabaja con algún objeto o dispositivo o que usa algún servicio en particular. - **Framework:** Plataforma para el desarrollo de aplicaciones de software que proporciona una base en la que los desarrolladores puedan	-1000 Usuarios. -2000 Usuarios. -3000 Usuarios. -Laravel. -Django.	-Usuarios concurrentes. -Componentes de Framework de lado del servidor.Haciendo uso de las tecnologías de php y python.

			crear programas. (Laravel, Django). - **Tipo de Operación:** Categoría de un proceso a realizarse en un sistema. (Consulta o inserción de registros)	-Inserción. -Consulta.	Utilizando MVC y MTV. -Uso del ORM. (Mapeo de objetos relacional) de cada framework para el manejo de los datos tanto como consulta e inserción.

(Elaboración propia)

1.5. Metodología de la Investigación

1.5.1. Tipo de Investigación

La investigación es de nivel descriptivo; (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014), definen un estudio descriptivo de la siguiente manera: “Es decir únicamente pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar como se relacionan estas”.

1.5.2. Diseño de la Investigación

La investigación según, Hernández Sampieri (2010), es experimental, ya que el primer requisito de un experimento es la manipulación intencional de una o más variables independientes, que se considera como supuesta causa en una relación entre variables y al efecto provocado por dicha causa se le denomina variable dependiente.

El diseño es factorial, dado que se va a analizar los efectos que tienen los factores o indicadores de las variables independientes como son la operación, la cantidad de usuarios que acceden a la aplicación Web, así como el marco de trabajo en sí.

El diseño factorial de esta investigación sería de 2x3x2 y la ecuación sería: $y_{A,B,C,m} = \mu + \alpha_A + \beta_B + \gamma_C + (\alpha\beta)_{AB} + (\alpha\gamma)_{AC} + (\beta\gamma)_{BC} + (\alpha\beta\gamma)_{ABC} + error$

Tabla 2. Factores y niveles de las variables independientes

Factor	Niveles
A: Marco de trabajo	Laravel
	Django
B: Cantidad de usuarios	1000 usuarios
	2000 usuarios
	3000 usuarios
C: Operación	Inserción
	Consulta

(Elaboración propia)

1.5.3. Métodos e instrumentos de recolección de datos

Para observar los rendimientos de las aplicaciones Web resultantes se utilizarán:

- La guía de observación de tiempo de ejecución: toma medidas del tiempo promedio de ejecución de la aplicación Web de un módulo Web utilizando los marcos de trabajo de lado servidor Laravel y Django.
- La guía de observación de uso de memoria RAM: toma medidas del tiempo uso promedio de memoria RAM de la aplicación Web de un módulo Web utilizando los marcos de trabajo de lado servidor Laravel y Django.

1.5.4. Tratamiento y Análisis de datos

Se utilizará el software estadístico SPSS mediante el cual se aplicarán pruebas de análisis de varianza o ANOVA y comparación de medias. También se hará uso del software Apache JMeter para medir el rendimiento general de las aplicaciones Web resultantes del uso de cada uno de los marcos de trabajo.

1.6. Justificación, importancia y beneficiarios de la investigación

1.6.1. Justificación de la investigación

El proyecto actual pretende analizar comparativamente los frameworks Django y Laravel debido a que exponen simplicidad al momento de desarrollar software y son populares en los lenguajes de programación Python y PHP. De acuerdo con los resultados de la Encuesta para Desarrolladores 2016 (Stack Overflow, 2016) PHP y Python se encuentran dentro de las 10 tecnologías más populares por los programadores fullstack, ubicándose en el 4to y 9no lugar respectivamente. (Ver Anexo 3)

Además, en la comparación de tendencias de búsqueda según Google Trends (2017), en su categoría por lenguaje de programación, se observa que el índice de búsquedas de estos frameworks en los últimos años es mayor en comparación a sus homólogos y coetáneos. Además, se puede observar en la comparación del último año que la tendencia entre Laravel y Django es similar con una ligera superioridad de este último. (Ver Anexo 4)

En la investigación se desarrollará un módulo por cada framework de una aplicación Web para el módulo Web más desarrollado por las empresas piuranas de acuerdo a la investigación de Ramírez Coronado (2017) que se corresponde con el de trámite documentario (Ver Anexo 5).

1.6.2. Importancia de la Investigación

Esta investigación representará un aporte para las empresas de desarrollo que no cuentan con la disponibilidad de tiempo para realizar este tipo de investigación, explorando en primera instancia de manera teórica, pero también práctica de los framework, para permitir a su vez un aporte teórico, para la comprensión, así como para encontrar las estructuras equivalentes, en función a la sintaxis, a comparar.

1.6.3. Beneficiarios de la Investigación

Desde desarrolladores freelance hasta empresas dedicadas al desarrollo de software que utilicen los framework Laravel o Django (basados en los lenguajes de programación PHP y Python, respectivamente) podrán tomar beneficio de esta investigación pues les dará una perspectiva más amplia de los mismos en escenarios de pruebas de estrés, así como visualizar que recursos y ventajas nos ofrece cada uno.

CAPITULO II

MARCO INSTITUCIONAL, TEÓRICO Y NORMATIVO.

2.1. Antecedentes relacionados con la investigación.

Estudio comparativo de los frameworks Ruby on Rails y Django para la implementación de un sistema informático de control y administración de network marketing (Guerrero Benalcázar, 2016)

Realizado por Rubén Ignacio Guerrero Benalcázar perteneciente a la Universidad Técnica Del Norte - Facultad De Ingeniería En Ciencias Aplicadas - Carrera De Ingeniería En Sistemas Computacionales (2016). En este proyecto se realizó una comparación entre los frameworks en base a los parámetros de aprendizaje, portabilidad, documentación, soporte, plantillas, vistas dinámicas, seguridad y sesiones a través de su valoración la cual fue entre el rango de 1 a 5. De esta manera se obtuvo los indicadores que determinaron la herramienta ganadora que fue Django el cual se utilizó en desarrollo de un sistema de Network marketing. Se llegaron a las siguientes conclusiones: La aplicación de una nueva metodología aplicada a la Web permite tener otro punto de vista en cuanto a tecnologías y metodologías se refiere, ya que se sale de los esquemas a los que se está acostumbrados. Esto ayudará a tener una mayor facilidad de comprensión de las aplicaciones Web y sus procedimientos.

Estudio y análisis de los framework en PHP basados en el modelo vista controlador para el desarrollo de software orientado a la web (Sierra, Acosta, Ariza, & Salas, s.f)

F. Sierra, J. Acosta, J. Ariza y M. Salas. Se tuvo como objetivo el indagar en el mundo de los frameworks en PHP, buscando información detallada sobre cada uno de estos, fecha de creación, usabilidad, entorno de diseño, sus componentes, realizando cuadros comparativos sobre las características de los distintos frameworks encontrados analizando los datos recopilados y obtener una visión de cuál de estos es más completo y más fácil de utilizar para el usuario. Se mostraron conceptos de algunos de estos frameworks y definiciones de algunas propiedades que estos contienen, también se realizaron algunos cuadros comparativos donde se resaltan ventajas, desventajas, fabricantes de estos, entre otros datos, dando a los usuarios una idea de cuál elegir para satisfacer sus necesidades, a la hora de realizar tareas con frameworks en lenguaje PHP. Conclusiones: Después de haber estudiado y analizado los frameworks de lado servidor para PHP, se concluyó que estos ahorran tiempo y tareas a la hora de un desarrollo web cualquiera que sea, ya que, por sus plantillas, complementos, compatibilidad y su forma de trabajo basado en el MVC, proporciona al usuario una mayor facilidad a la hora de el desarrollo de una aplicación.

Análisis comparativo del tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo en el lado cliente y en el lado servidor

Ramírez Coronado (2017), formula ¿Cuál es el tiempo de ejecución de una aplicación Web que ha sido desarrollada con diferentes marcos de trabajo utilizados en el lado del cliente y en el lado del servidor? Además, plantea como objetivo Realizar un análisis comparativo del tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo utilizados en el lado del cliente y en el lado del servidor. La investigación se realizó con la finalidad de determinar qué marcos de trabajo permiten que la aplicación a desarrollar tenga mejores tiempos de ejecución, debido a que estos deben ser cortos. Como hipótesis se planteó Existe diferencia significativa en el tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo utilizados en el lado del cliente y en el lado del servidor.

2.2. Framework

En TechTerms (2013), se define un framework como “una plataforma para el desarrollo de aplicaciones de software que proporciona una base en la que los desarrolladores puedan crear programas para una plataforma específica”.

Un framework o marco de trabajo “ofrece componentes como librerías, plantillas o esqueletos que definen el funcionamiento de las aplicaciones permitiendo manejar y controlar prácticamente la mayoría de la mayor parte de la misma sin escribir mucho código” GNUSTEP (s.f.).

Uno de los frameworks más populares en la actualidad son Laravel y Django basados en los lenguajes de programación PHP y Python, que a su vez se encuentran entre los diez lenguajes de programación más populares por parte de desarrolladores full-stack en la encuesta de desarrolladores efectuada por Stack Overflow (2016).

De manera general un framework ayuda a acelerar la velocidad de desarrollo de un proyecto, cuenta con un código base optimizado, permite una mejor colaboración entre los integrantes de un mismo grupo de desarrollo al contar con estándares y convenciones de código.

2.3. Marco de trabajo Laravel

McCool (2012), define Laravel como “un marco de trabajo de desarrollo web escrito en PHP. Ha sido diseñado para mejorar la calidad del software reduciendo el costo de desarrollo inicial como los de mantenimiento continuo, asi como para mejorar la experiencia al trabajar en las aplicaciones al proporcionar una sintaxis clara y funcionalidades que ayudarán a ahorrar horas de tiempo de implementación.”

2.3.1. Características

Bean (2015), indica que este framework aumenta la productividad ya que tiene las siguientes caracteristicas:

- **Modularidad:** Laravel fue construido sobre más de 30 librerías diferentes y se divide en módulos individuales. Asimismo, está integrado con el gestor de dependencias Composer, el cual ayuda a actualizar los componentes que usa con facilidad
- **Verificabilidad:** este framework fue construido desde cero para facilitar las pruebas haciendo uso de helpers que le permitan acceder a las rutas desde dichas pruebas, rastrear el código html resultante, asegurándose que se utilicen las clases correctas en el momento adecuado.
- **Ruteo:** Laravel brinda mucha flexibilidad para poder definir las rutas de la aplicación. Por ejemplo, se puede enlazar manualmente una función anónima a una ruta con el verbo http como get, post, put o delete. Esta funcionalidad fue inspirada por micro frameworks como Sinatra (Ruby) y Silex (PHP).
- **Gestión de configuración:** Laravel tiene un enfoque consistente para manejar diferentes ajustes de configuración los cuales pueden ser aplicados a varios entornos a través del archivo .env.
- **Query Builder y ORM:** Laravel posee un generador de consultas (Query builder) que le permite realizar, a través de métodos que

siguen la sintaxis de PHP, emitir consultas hacia la base de datos sin escribir sintaxis SQL. Adicionalmente, cuenta con un Mapeador de objetos relacionales (ORM) llamado Eloquent, el cual utiliza el patrón Active Record. Ambas formas de conexión hacia la base de datos son compatibles para diferentes sistemas gestores de base de datos como PostgreSQL, SQLite, MySQL y SQL Server.

- **Migraciones:** esta funcionalidad está inspirada en Rails, la cual permite definir el esquema de la base de datos utilizando clases de PHP, así como tener un seguimiento de los cambios realizados en dichos esquemas.
- **Motor de plantillas:** Laravel utiliza Blade, un lenguaje de plantilla ligero con el cual se puede diseños jerárquicos con bloques de contenido los cuales pueden ser inyectados de manera dinámica.
- **Autenticación:** dado que la autenticación de usuarios es una funcionalidad común en aplicaciones web, este framework tiene una implementación por defecto para registrar, autenticar e incluso recordar las contraseñas.

2.3.2. Instalación

En Laravel (2019), se indica que este marco de trabajo tiene algunos requerimientos para su instalación, los cuales son abarcados por la máquina virtual Laravel Homestead. Sin embargo, si no se usa esta herramienta se debe corroborar que el servidor cumpla con la siguiente lista de extensiones y versiones:

- PHP >= 7.1.3
- BCMath PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

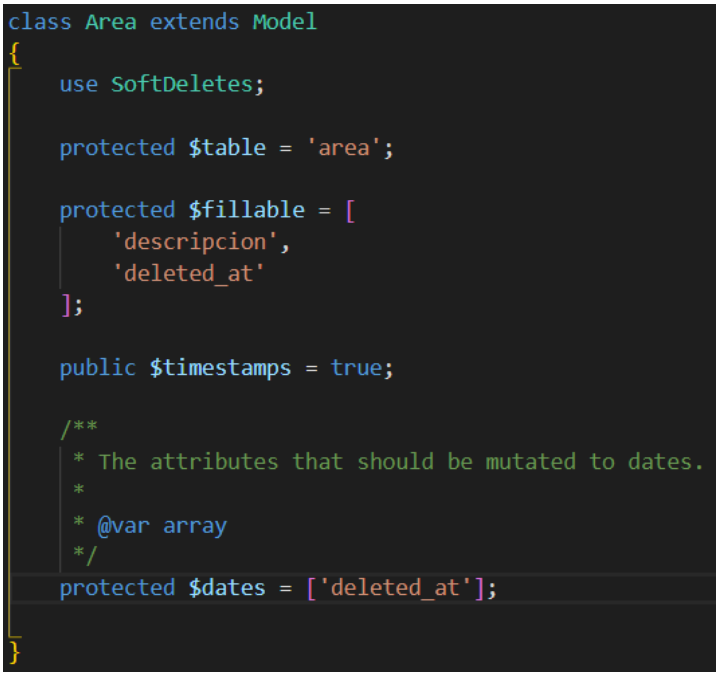
Laravel utiliza Composer para manejar las dependencias que utiliza así que es necesario instalarlo. Para poder crear un proyecto con este framework se debe escribir en la terminal el siguiente comando: *composer create-project --prefer-dist laravel/laravel nombre_proyecto*.

2.3.3. Estructura

Bean (2015), indica que los elementos básicos para poder construir una aplicación con este framework son los siguientes:

- **Modelos:** los modelos representan registros de una base de datos. Es decir se puede pensar en los modelos como entidades de la aplicación como un usuario, un articulo, etc. En este framework los modelos extienden de una clase base llamada Model y son nombrados usando la convención camel case (por ejemplo NewArticle).

Ilustración 1. Estructura de un modelo en Laravel



```
class Area extends Model
{
    use SoftDeletes;

    protected $table = 'area';

    protected $fillable = [
        'descripcion',
        'deleted_at'
    ];

    public $timestamps = true;

    /**
     * The attributes that should be mutated to dates.
     *
     * @var array
     */
    protected $dates = ['deleted_at'];

}
```

Fuente: Elaboración propia

- **Controladores o rutas:** los controladores en su forma más simple toman una solicitud, realizan una determinada acción y devuelven una respuesta adecuada. Se encargan del procesamiento de datos ya sea devolver información de una base de datos, manejar el envío de datos a través de un formulario y guardarlos.

Ilustración 2. Estructura de un controlador en Laravel

```php
class UsersController extends Controller
{
    public function index(User $user)
    {
        return UsersResource::collection($user->paginate())->hide(['id', 'email']);
    }

    public function show(User $user)
    {
        return UsersResource::make($user)->hide(['id']);
    }
}
```

Fuente: Elaboración propia

- **Vistas o plantillas:** las vistas son responsables de mostrar la respuesta enviada por el controlador en un formato adecuado, usualmente en una página HTML. Pueden ser construidas usando el lenguaje de plantillas Blade o simplemente usando PHP estándar.

Ilustración 3. Estructura de una vista con Blade

app.blade.php

```
<!doctype html>
<html @php language_attributes() @endphp>
  @include('partials.head')
  <body @php body_class() @endphp>
    @php do_action('get_header') @endphp
    @include('partials.header')
    <div class="wrap container" role="document">
      <div class="content">
        <main class="main">
          @yield('content')
        </main>
        @if (App\display_sidebar())
          <aside class="sidebar">
            @include('partials.sidebar')
          </aside>
        @endif
      </div>
    </div>
    @php do_action('get_footer') @endphp
    @include('partials.footer')
    @php wp_footer() @endphp
  </body>
</html>
```

Fuente: Elaboración propia

2.4. Marco de trabajo Django

Django es un framework web extremadamente popular y completamente funcional, escrito en Python. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago.

En el artículo "*Django: A Python-Powered Development Framework*" (2016), se menciona que, en su núcleo, este es un framework de alto nivel de estilo MVC, que tiene una colección de bibliotecas escritas en Python, y uno de los más populares frameworks del lado servidor. Su lema es "No repitas". Al igual que Python, hace hincapié en la eficiencia, lo que le permite hacer todo lo posible con la menor codificación posible.

2.4.1. Patrón MTV

Django sigue el patrón MVC tan al pie de la letra que puede ser llamado un framework MVC. Someramente, la M, V y C se separan en Django de la siguiente manera:

- M, la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django.
- V, la porción que selecciona qué datos mostrar y como mostrarlos, es manejada por la vista y las plantillas.
- C, la porción que delega a la vista dependiendo de la entrada de usuario, es manejada por el framework mismo siguiendo tu URLconf y llamando a la función apropiada de Python para la URL obtenida.

Debido a que la "C" es manejada por el mismo framework y la parte más importante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework MTV.

- M significa "Model" (modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, como validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

- T significa “Template” (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas mostradas sobre una página web otro tipo de documento.
- V significa “View” (Vista), la capa de la lógica, de negocios. Esta capa contiene la lógica del modelo y la delega a la plantilla apropiada, se puede pensar en esto como un puente entre los modelos y las plantillas.

2.4.2. Características

Enrech Enrech (2013), menciona alguna de las características más importantes:

- **Mapeador objeto-relacional:** las clases del modelo (base de datos) se definen en Python y se trabaja con la API de acceso a la base de datos que provee Django. Esto permite desarrollar independientemente el motor de la base de datos y evita en cierta medida el uso de SQL.
- **Vistas genéricas:** incorpora un sistema de vistas genéricas para hacer tareas habituales: listar registros, ver el detalle de un registro, borrar un registro, etc.
- **URL’s elegantes:** permite crear URL’s elegantes y limpias haciendo servir expresiones regulares.
- **Sistema de plantillas:** incorpora un sistema de plantillas que permite separar diseño gráfico y programación. Se puede editar el HTML sin tocar el código Python.
- **Cache:** usa memcached2 para obtener buen rendimiento.
- **Aplicaciones *pluggables*:** las aplicaciones se pueden instalar en cualquier otro proyecto Django, es decir, son reutilizables.

2.4.3. Instalación

Para poder trabajar con este framework primero se debe instalar Python (de preferencia la versión 3), posteriormente se trabajó con una dependencia llamada virtualenv, encargada de crear entornos virtuales y aislar todos los recursos que se usan en cada proyecto, siendo además reutilizables y que permite trabajar con distintas versiones tanto del framework como de las distintas dependencias que se necesiten.

Para proceder con la instalación se usa el gestor de paquetes que trae Python por defecto, el cual es pip utilizando el comando: *pip install virtualenv*.

Es recomendable crear una carpeta en donde se coloquen todos los entornos virtuales los que se trabajarán. Para poder crear un entorno se utiliza el comando: *virtualenv nombre_entorno*.

Tras ejecutar el comando anterior se crea una carpeta con el nombre del entorno. Usando la terminal de comandos se accede a la carpeta Scripts del entorno creado, y se escribe la palabra activate, la cual hará que se active el entorno virtual, se puede comprobar que es así ya que el nombre del entorno aparecerá al inicio de la ruta actual de la terminal.

Con el entorno activado se puede ejecutar el comando *pip install django* el cual instala el framework.

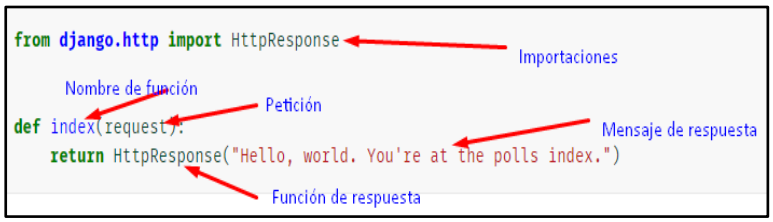
2.4.4. Estructura

2.4.4.1. Vistas

Una vista es la que contiene la lógica de nuestra aplicación, podemos crearlas de 2 maneras:

a) Basadas en función

Ilustración 4. Vista basada en función



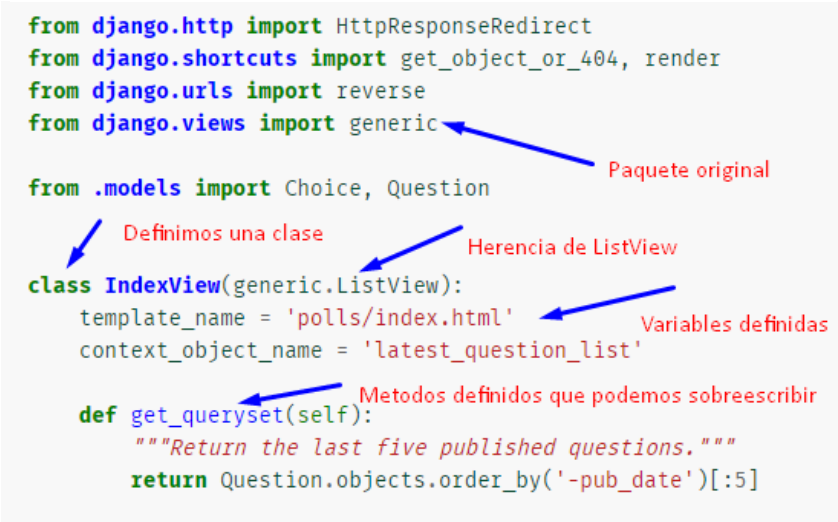
Fuente: Elaboración propia basada en documentación de Django

Esta es una vista muy sencilla, la cual sólo muestra un mensaje en el navegador, pero básicamente así es la estructura de las vistas basadas en funciones. Se utiliza código puro de Python, sin usar funciones definidas o ya creadas en Django.

b) Basadas en clases

Las vistas genéricas abstraen los patrones comunes hasta el punto en que ni siquiera es necesario escribir código Python para escribir una aplicación.

Ilustración 5. Vista basada en clases

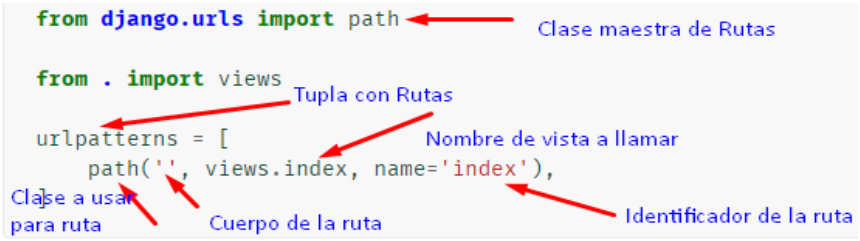


Fuente: Elaboración propia basada en documentación de Django

2.4.4.2. Rutas

Las vistas son llamadas por las Rutas o urls, la estructura de estas es:

Ilustración 6. Uso de rutas - forma 1



Fuente: Elaboración propia basada en documentación de Django

Las rutas no tienen una sola estructura, también pueden ser:

Ilustración 7. Uso de rutas - forma 2

Fuente: Elaboración propia basada en documentación de Django

Donde la función include() permite hacer referencia a otros URLconfs. Cada vez que Django encuentra include() corta cualquier parte de la URL que coincide hasta ese punto y envía la cadena restante a la URLconf incluida para seguir el proceso. La idea detrás de include() es facilitar la conexión y ejecución inmediata de las URLs.

2.4.4.3. Modelos

Un modelo es la fuente única y definitiva de información sobre los datos. Contiene los campos esenciales y los comportamientos de los datos que se deben guardar. Django sigue el Principio DRY cuyo objetivo es definir el modelo de datos en un solo lugar y derivar cosas de este automáticamente.

Ilustración 8. Estructura de un modelo en DJango

```
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

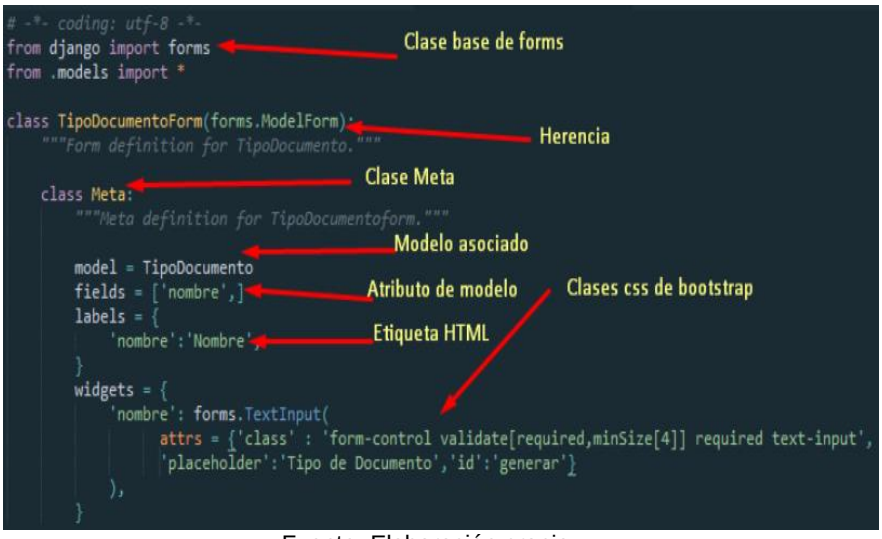
Fuente: Elaboración propia basada en documentación de Django

2.4.4.4. Formularios

Holovaty & Kaplan Moss (2008) definen un formulario de Django como “una subclase de django.newforms.Form, tal como un modelo de Django es una subclase de django.db.models.Model. El módulo django.newforms también contiene cierta cantidad de clases Field para los campos”.

La estructura básica de un formulario en este framework se muestra en la Ilustración 9.

Ilustración 9. Estructura de un formulario en Django



Fuente: Elaboración propia

2.5. Análisis de Marcos de Trabajo: Laravel vs Django

Tomando la información encontrada en SimilarTech (2019) así como la que fue mostrada en la sección anterior se decidió elaborar un pequeño cuadro comparativo general de ambos frameworks.

Tabla 3. Comparativa Laravel y Django

Aspecto de comparación	Django	Laravel
Uso en sitios web reconocidos	A pesar de su reciente crecimiento, Django aún está detrás de Laravel en todos los segmentos de participación de mercado considerado por SimilarTech (2019)	Laravel es líder en uso en los 10mil y 100mil mejores sitios web con un 0.8% y 1.061% respectivamente. (SimilarTech, 2019)
Geografía	Django es líder (respecto a Laravel) en Estados Unidos, Rusia, Francia, Bielorrusia y otros 26 países.	Laravel es el líder (respecto a Django) en la mayoría de países incluyendo Brasil, Inglaterra, China, India y otros 174 países.
Patrón por el que es más reconocido	MTV	MVC
Estructura	Modelo Controlador Rutas Vista o plantilla	Vistas Rutas Modelos Formularios y plantillas

Fuente: Elaboración propia

Por otro lado, también se decidió comparar las estructuras que utilizan cada framework tratando de buscar las equivalencias para cada par mostrado.

Los modelos en cada framework son utilizados de manera diferente, en Laravel se crear una clase (y archivo) por cada tabla en la base de datos por ejemplo si se trabajara con las tablas persona y unidad, en este framework se tendría que crear dos clases Persona.php y Unidad.php. En Django es totalmente diferente: la estructura de la aplicación maneja un archivo llamado models.py dentro del cual se definen las clases que corresponden a las tablas de la base de datos. Tomando como ejemplo el

caso anterior también se trabajaría con dos clases, salvo que ambas estarían dentro del mismo archivo.

Ilustración 10. Modelo Persona y UnidadNegocio - Framework Laravel

```php
class UnidadNegocio extends Model
{
    use SoftDeletes;

    protected $table = 'unidad_negocio';

    protected $fillable = [
        'descripcion',
        'usuario',
        'pc',
        'ip',
        'usuarioMod',
        'ipMod',
        'pcMod',
        'deleted_at'
    ];

    public $timestamps = true;

    /**
     * The attributes that should be mutated to dates.
     *
     * @var array
     */
    protected $dates = ['deleted_at'];
```

```php
class Persona extends Model
{
    use SoftDeletes, BaseTrait;

    protected $table = 'persona';

    protected $appends = ['descripcion'];

    protected $fillable = [
        'razonSocial',
        'nombres',
        'apellidos',
        'numeroDocumento',
        'direccion',
        'tipoPersonaId',
        'tipoEntidadId',
        'tipoContribuyenteId',
        'documentoIdentidadId',
    ];

    public $timestamps = true;

    /**
     * The attributes that should be mutated to dates.
     *
     * @var array
     */
    protected $dates = ['deleted_at'];
```

Fuente: Elaboración propia

Ilustración 11. Modelo TipoDocumento y TipoTramite - Framework Django

models.py

apps ▸ tramite ▸ models.py

```python
# -*- coding: utf-8 -*-
from django.db import models
from .validators import validate_even
from apps.usuarios.models import Unidad,Persona,User,PersonaNatural

class TipoDocumento(models.Model):
    """Model definition for TipoDocumento."""
    # TODO: Define fields here
    id = models.AutoField(primary_key = True)
    nombre = models.CharField('Nombre de Tipo de Documento', max_length=255,unique = True)
    estado = models.CharField('Tipo de Documento',max_length=1,default='1')
    deleted_at = models.TimeField('Deleted', auto_now = False, auto_now_add = True)
    created_at = models.TimeField('Created', auto_now = False, auto_now_add = True)
    updated_at = models.TimeField('Updated', auto_now = False, auto_now_add = True)

class TipoTramite(models.Model):
    """Model definition for TipoTramite."""
    # TODO: Define fields here
    id = models.AutoField(primary_key = True)
    nombre = models.CharField('Nombre de Tipo de Tramite', max_length=255,unique = True, validators = [validate_even])
    deleted_at = models.TimeField('Deleted', auto_now = False, auto_now_add = True)
    created_at = models.TimeField('Created', auto_now = False, auto_now_add = True)
    updated_at = models.TimeField('Updated', auto_now = False, auto_now_add = True)
```

Fuente: Elaboración propia

Django utiliza vistas o views para poder capturar una petición web y devolver una respuesta, esto incluye retornar un template o un JSON. En Laravel existe un elemento que se encarga de lo mismo, pero se le denomina controlador. El uso de ambos elementos para cada framework

se maneja como con los modelos: Django posee un archivo views.py en el cual se definen todas las clases o vistas, mientras que Laravel lo maneja de manera independiente siempre y cuando cada controlador extienda de la clase Controller.

Ilustración 12. Controller AreaController - Framework Laravel

```php
class AreaController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function getList()
    {
        try
        {
            $listaAreas = Area::with('unidadNegocio')
                              ->withTrashed()
                              ->get();

            $listaAreas->map(function ($item) {
                $item->makeHidden('unidadNegocio');
                return $item->nombreUnidadNegocio = $item->unidadNegocio->descripcion;
            });

            return response()->json(['success' => true,'message' => $listaAreas,]);
        }
        catch(Exception $ex)
        {
            return response()->json(['success' => false,'message' => $ex->getMessage(),]);
        }
    }
```

Fuente: Elaboración propia

Ilustración 13. Vista BusquedaTramite - Framework Django

```python
views.py
apps ▸ tramite ▸ views.py
from openpyxl.styles import PatternFill
from openpyxl.styles import Side

#################### TIPO DOCUMENTO ######################

class BusquedaTramite(TemplateView):
    def get(self,request,*args,**kwargs):
        queryset = Tramite.objects.filter(persona_id = Persona.objects.filter(id = 1))
        lista_tipo_docu = []
        for tramite in queryset:
            docu_json = {}
            docu_json['id'] = tramite.id
            docu_json['codigo'] = tramite.codigo
            docu_json['dias_atencion'] = tramite.dias_atencion
            docu_json['folio'] = tramite.folio
            docu_json['asunto'] = tramite.asunto
            lista_tipo_docu.append(docu_json)
        data = json.dumps(lista_tipo_docu)

        return HttpResponse(data, 'application/json')
```

Fuente: Elaboración propia

Hasta ahora se podría decir que Laravel maneja sus elementos de manera separada, mientras que Django los utiliza en un solo archivo. Al definir las rutas de acceso para la aplicación ambos frameworks utilizan un solo

archivo, sin embargo, Laravel tiene dos archivos por defecto para poder colocarlas: uno si la aplicación está orientada a ser un api (api.php) o si se desea utilizar Laravel como un todo (web.php). Dentro de la estructura que se utilizó para Django solo existe el archivo urls.py.

Ilustración 14. Archivo de rutas api.php - Framework Laravel

api.php

routes ▸ api.php ▸ ...

```php
*/

Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::prefix('familias')->group(function () {
    Route::get('/', 'FamiliaController@getList');
    Route::get('/{familia}', 'FamiliaController@show');
    Route::post('/create', 'FamiliaController@store');
    Route::put('/{familia}', 'FamiliaController@update');
    Route::delete('/{familia}', 'FamiliaController@destroy');
    Route::get('/restore/{id}', 'FamiliaController@restore');
});
```

Fuente: Elaboración propia

Ilustración 15. Archivo de rutas urls.py - Framework Django

urls.py

apps ▸ tramite ▸ urls.py

```python
from django.conf.urls import url
from .views import *
from django.contrib.auth.decorators import login_required

urlpatterns = [
############################ TIPO DOCUMENTO ##############################
    url(r'^crear_tipo_documento/', login_required(TipoDocumentoCreateView.as_view()), name='crear_tipo_documento'),
    url(r'^listar_tipo_documento/', login_required(TipoDocumentoListView.as_view()), name='listar_tipo_documento'),
    url(r'^update_tipo_documento/(?P<pk>\d+)/$', login_required(TipoDocumentoUpdateView.as_view()), name='update_tipo_documento'),
    url(r'^delete_tipo_documento/(?P<pk>\d+)/$', login_required(TipoDocumentoDeleteView.as_view()), name='delete_tipo_documento'),
    url(r'^tipo_documento/', login_required(CrearTipoDocumento), name='tipo_documento'),
    url(r'^buscar_tramite/', BusquedaTramite.as_view(), name='buscar_tipo_documento'),
############################ TIPO TRAMITE ##############################
```

Fuente: Elaboración propia

Finalmente, para poder mostrar una pantalla o formulario utilizando estos frameworks, ambos trabajan de manera totalmente diferente. Laravel utiliza archivos que trabajan con el sistema de plantillas Blade, que permite generar código html dinámico con una sintaxis mucho más limpia que si se usara php plano. Estos archivos trabajan con la extensión blade.php.

Django por su parte trabaja mediante dos elementos: formularios y plantillas. Un proyecto que utilice este framework puede configurarse con uno o varios motores de plantillas. Django tiene su propio sistema de plantillas, y con el cual se trabajó para esta investigación, llamado lenguaje de plantillas Django (DTL). Dentro de su estructura se ubica una carpeta llamada templates dentro de la cual se colocan archivos con extensión html sin embargo dentro de este se puede agregar código de DTL. En el archivo forms.py se define la estructura de todos los formularios con los que trabaja la aplicación, definiendo etiquetas, validaciones, etc.

Ilustración 16. Plantilla index - Framework Laravel

index.blade.php

resources ▸ views ▸ index.blade.php

```
17    <!-- START PAGE CONTAINER -->
18        <div class="page-container">
19
20            <!-- START PAGE SIDEBAR -->
21            <div class="page-sidebar">
22                <!-- START X-NAVIGATION -->
23                <ul class="x-navigation">
24                    <li class="xn-logo">
25                        <a>{{ auth()->user()->name }}</a>
26                        <a href="#" class="x-navigation-control"></a>
27                    </li>
```

Fuente: Elaboración propia

Ilustración 17. Archivo forms.py – Framework Django

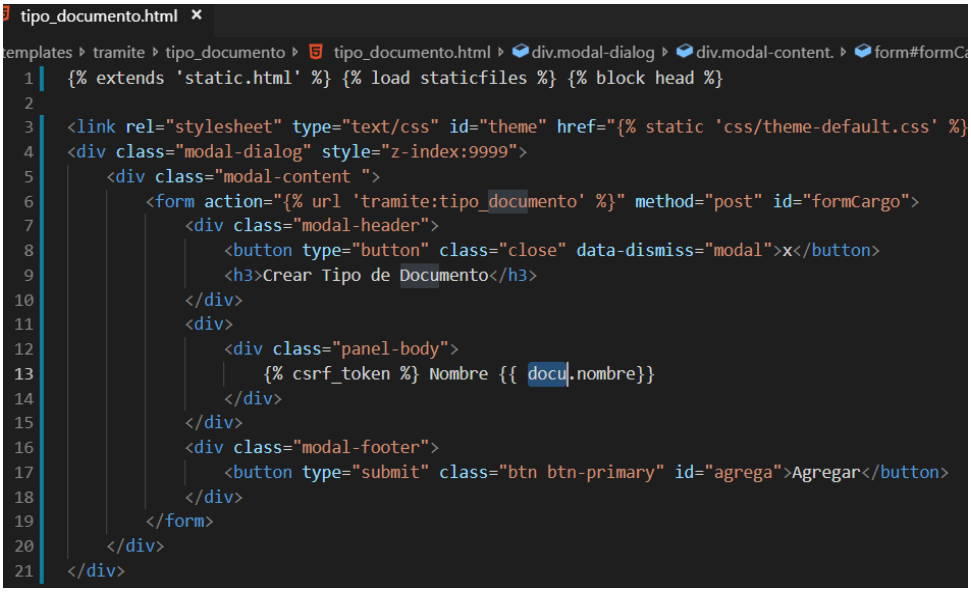
forms.py

apps ▸ tramite ▸ forms.py

```
1   # -*- coding: utf-8 -*-
2   from django import forms
3   from .models import *
4
5   class TipoDocumentoForm(forms.ModelForm):
6       """Form definition for TipoDocumento."""
7
8       class Meta:
9           """Meta definition for TipoDocumentoform."""
10
11          model = TipoDocumento
12          fields = ['nombre',]
13          labels = {
14              'nombre':'Nombre',
15          }
16          widgets = {
17              'nombre': forms.TextInput(
18                  attrs = {'class' : 'form-control valid
19              ),
20          }
```

Fuente: Elaboración propia

Ilustración 18. Plantilla tipo_documento - Framework Django



Fuente: Elaboración propia

Para mayor detalle del código utilizado en este proyecto para cada framework ver el Anexo 6.

2.6. Apache JMeter

"Es una aplicación open source desarrollada en Java, diseñada para ejecutar pruebas de funcionalidad y rendimiento. Originalmente se creó para ejecutar pruebas en aplicaciones web, pero desde se ha expandido sus funcionalidades". JMeter (2019)

2.7. Pruebas de aplicaciones web

Las pruebas de software se realizan con el fin de asegurar la calidad y la seguridad de los productos que se ofrecen a los clientes.

"Ya que los sistemas web residen en red y son accesibles desde muchos sistemas operativos, navegadores de varios dispositivos, etc, la búsqueda de errores representa un reto significativo. Las pruebas se enfocan en

contenido, función, estructura, usabilidad, navegabilidad, rendimiento, compatibilidad, interacción, capacidad y seguridad. Estas pruebas ocurren mientras se diseña y desarrolla la aplicación y pruebas que se llevan a cabo una vez que se implementa". Villegas Avilés (2012).

En esta investigación se centró en un aspecto: rendimiento, tomando en cuenta dos indicadores: tiempo de respuesta y uso de memoria RAM.

2.7.1. Pruebas de rendimiento

Las pruebas de rendimiento web se ejecutan al navegar a través de la aplicación web. Este tipo de pruebas se incluyen en otras denominadas de carga, para poder medir el rendimiento de una aplicación cuando es utilizada por varios usuarios. Estas pruebas pueden convertirse en un script que se puede editar y personalizar como cualquier otro programa. (Microsoft Developer Network, 2016).

En esta investigación en particular se utilizó una herramienta (Apache JMeter) para poder ejecutarlas, pues facilitan la generación y recolección de datos tras poner en ejecución la prueba.

a) Tiempo de respuesta

El tiempo de respuesta es uno de los indicadores que se usan para determinar el rendimiento que pueda tener una aplicación

frente a una determinada carga de trabajo. El tiempo de respuesta puede ser definido como “el lapso de tiempo entre el fin de una petición en un sistema informático y el inicio de la recepción de la respuesta” (IBM, 1993).

2.8. Pruebas estadísticas

Las pruebas estadísticas permiten el análisis de los datos obtenidos de las diferentes pruebas con las que se cuentan al momento de la ejecución de una investigación, para el estudio de los datos obtenidos en el desarrollo de la presente investigación se empleó la siguiente prueba:

2.8.1. Análisis de varianza factorial univariante - ANOVA

El análisis de varianza “permite contrastar la hipótesis de igualdad de medias de las poblaciones definidas por los diferentes niveles en que se puede segmentar un factor o variable dependiente” (Bakieva, Gonzáles Such, & Jornet, 2015)

Cuando se desea estudiar el efecto de más de un factor sobre la variable dependiente, es preciso recurrir a los modelos factoriales de análisis de varianza que permitan estudiar el efecto de diversos factores, tanto de manera individual como conjunta.

En este caso se tomó en cuenta 3 factores, por lo que en el análisis ANOVA se deben considerar: “los efectos de cada factor por separado sobre la variable dependiente, que se conocen como efectos principales, y el efecto de la interacción de ambos factores sobre la variable

dependiente. Si el número de factores fuera tres, los efectos a estudiar serían siete (tres principales, tres interacciones dobles y una interacción triple). Si el número de factores fueran cuatro, los efectos a estudiar serían 15 (cuatro principales, seis interacciones binarias, cuatro interacciones tripes y una interacción cuádruple), y así sucesivamente". (MLG. ANOVA factorial univariante)

CAPITULO III

MARCO METODOLÓGICO

3.1. Enfoque y diseño

Esta investigación fue desarrollada bajo un enfoque cuantitativo, siendo el diseño de la investigación experimental ya que se evaluó el rendimiento en cuanto a tiempo de respuesta y uso de memoria RAM bajo dos tipos de operación (registro y consulta), número de usuarios y frameworks.

Es de nivel descriptivo pues su objetivo es medir el efecto en el tiempo de respuesta y uso de memoria RAM, debido a las variaciones del número de usuarios, el tipo de operación y frameworks.

(Hernández Sampieri, Fernández Collado, & Baptista Lucio, Metodología de la Investigación, 2014), definen un estudio descriptivo de la siguiente manera: “Con frecuencia, la meta del investigador consiste en describir fenómenos, situaciones, contextos y sucesos; esto es detallar como son y se manifiestan. Con los estudios descriptivos se busca especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis. Es decir, únicamente pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar como se relacionan estas.”

3.2. Sujeto de la investigación

Esta investigación no contó con sujetos de investigación, en su lugar se contó con elementos de investigación, siendo representados por los tiempos de respuesta y uso de memoria RAM, cuyos valores se obtuvieron

al realizar las pruebas de rendimiento en los módulos desarrollados en los frameworks Django y Laravel.

3.3. Métodos y procedimientos

3.3.1 Desarrollo del módulo web

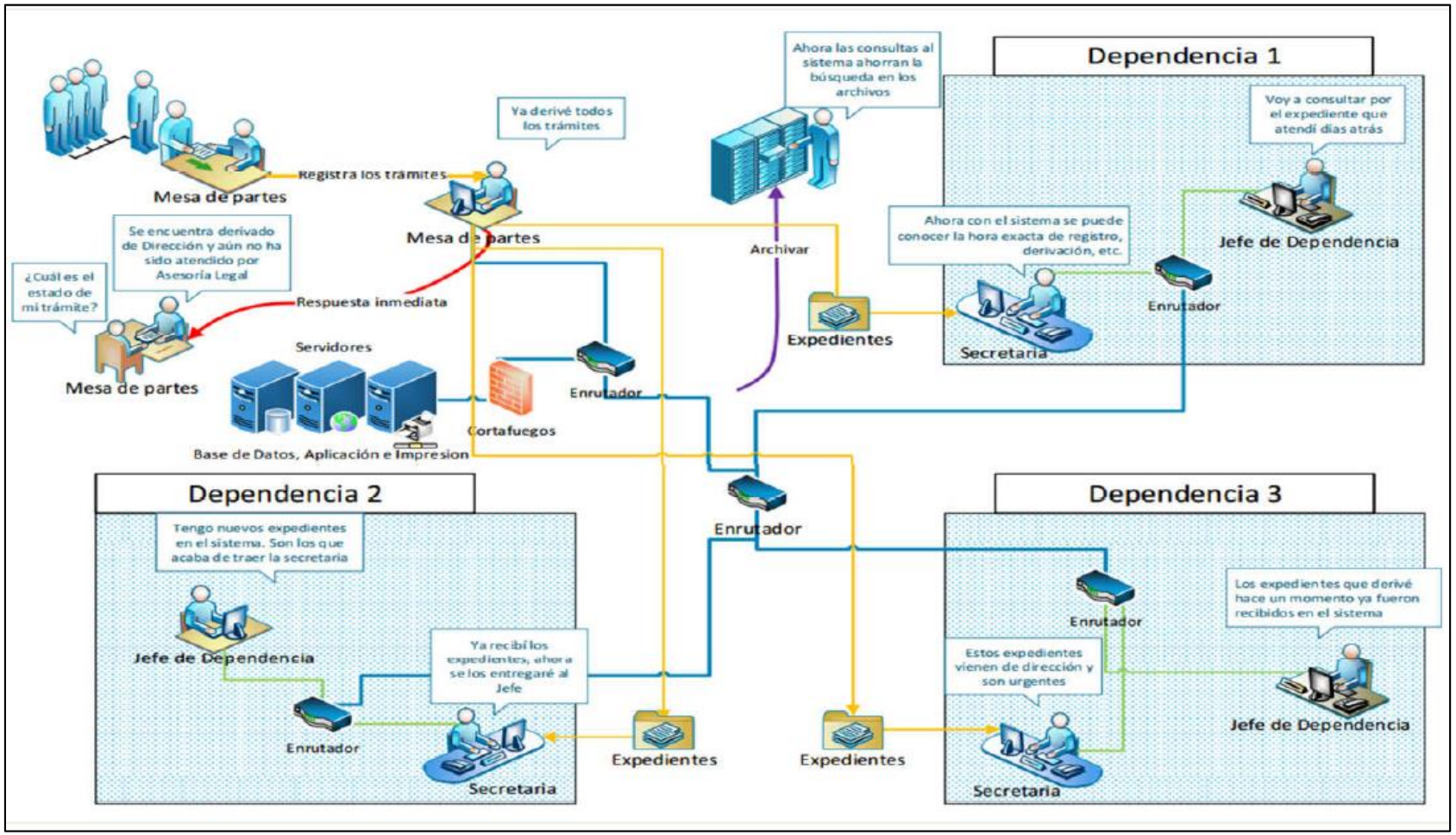
Para poder determinar que módulo web desarrollar, se tomó en cuenta la tesis “Análisis comparativo del tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo en el lado cliente y en el lado servidor. Piura” presentada por Ramírez Coronado (2017). En ella se concluye que el módulo más desarrollado o utilizado por las empresas de esta región es el de trámite documentario representado por el 31% de los encuestados.

Para poder comprender el funcionamiento o requerimientos de este módulo se tomó como referencia la tesis “Sistema informático web de trámite documentario para la Ugel de Zarumilla – Tumbes utilizando los frameworks AngularJS Y Spring MVC” desarrollada por Calmet Izquierdo (2014), en la cual se muestra el flujo de funcionamiento de este tipo de sistemas como se muestra en la *Ilustración 19.*

Las dos tesis mencionadas fueron complemento para poder establecer los requerimientos funcionales, modelado de negocio, procesos, etc, que se abarcarían para esta investigación.

En este sentido, la presente investigación delimita el análisis comparativo a un módulo de trámite documentario.

Ilustración 19. Proceso de trámite documentario en una organización



Fuente: Calmet Izquierdo (2014)

3.3.1.1 Requerimientos

En la tabla se listan los requerimientos conciernentes al módulo de trámite documentario que fueron considerados por Ramírez Coronado (2017):

Tabla 4. Requerimientos funcionales - módulo de trámite documentario

Item	Requerimiento
RF01	El sistema permitirá a los usuarios consultar la bandeja de expedientes finalizados
RF02	El sistema permitirá a los usuarios registrar un trámite ya sea de tipo externo o interno.

Fuente: Ramírez Coronado (2017)

3.3.1.2 Especificación de los requerimientos

Tabla 5. Especificación del requerimiento RF01

Consulta de bandeja de expedientes finalizados	
Actor	Mesa de Parte, Jefe de Unidad
Descripción	El sistema permite a los usuarios consultas los expedientes finalizados.
Flujo básico	Los usuarios ingresan a la opción bandeja de finalizados, se muestra una pantalla donde se listan los expedientes finalizados.
Flujos alternos	**No se han finalizados trámites** El sistema muestra en pantalla un mensaje indicando que no hay expedientes finalizados.

Pre - condiciones	Los usuarios deben haber ingresado a la opción bandeja finalizados. El usuario debe haber finalizado expedientes.
Post - condiciones	El sistema muestra una lista de los expedientes finalizados.

Fuente: Ramírez Coronado (2017)

Tabla 6. Especificación del requerimiento RF02

Registrar trámite	
Actor	Mesa de parte, Jefe de Unidad
Descripción	El sistema permitirá registrar un trámite ya se de tipo externo o interno
Flujo básico	Los usuarios acceden a la opción registrar tramite, se muestra una pantalla donde sebe indicar el tipo de trámite, remitente o solicitante, folios, asunto, acción, unidad y prioridad. El usuario llena todos los datos y presiona el botón de registrar
Flujos alternos	**Solicitante o remitente no se encuentra registrado** El sistema muestra un mensaje indicando que no se encuentra el solicitante buscado
Pre - condiciones	El usuario debe haber accedido a la opción registrar expediente
Post - condiciones	El sistema debe mostrar de confirmación indicando que el trámite se ha registrado correctamente.

Fuente: Ramírez Coronado (2017)

3.3.1.3. Diagramas

3.3.1.3.1. Diagrama de Negocio

Martín Díaz (2006), define este término como una técnica para modelar el funcionamiento de una organización a través de sus procesos de negocios.

Ramirez Coronado (2017), abarca este modelado en términos de casos de uso y actores. A continuación se listan los actores de negocio encontrados para el módulo de trámite documentario:

Tabla 7. Actores de negocio - módulo de trámite documentario

Nombre	Descripción
Solicitante	Persona natural o jurídica, que se acerca a la entidad a realizar cualquier trámite. Del mismo modo, las unidades pueden comportarse como solicitante al emitir un pedido a otra unidad, a esto se le considera trámite interno.
Mesa de partes	Usuario encargado de la recepción, registro y derivación de expediente.
Jefe de Unidad	Usuario de gestionar los expedientes que recibe en su unidad.

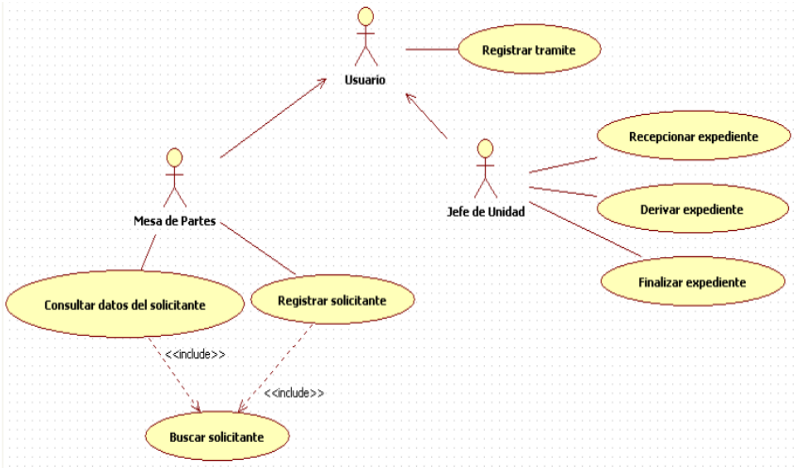
Fuente: Ramírez Coronado (2017)

3.3.1.3.2. Diagramas de casos de uso

Ramírez Coronado (2017), elaboró los casos de uso dividiendo los procesos o escenarios de la siguiente manera:

a) Gestión de expedientes

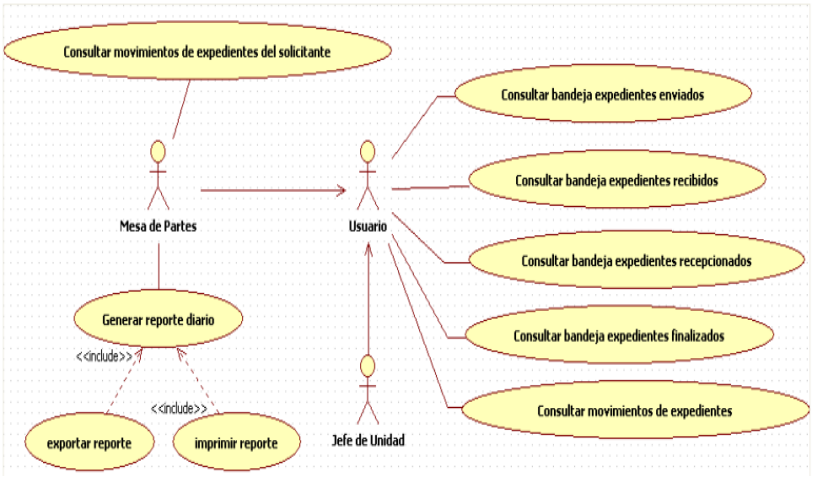
Ilustración 20. Casos de uso - Proceso de gestión de expedientes



Fuente: Ramírez Coronado (2017)

b) Control y monitoreo de expedientes

Ilustración 21. Casos de uso - Control y monitoreo de expedientes

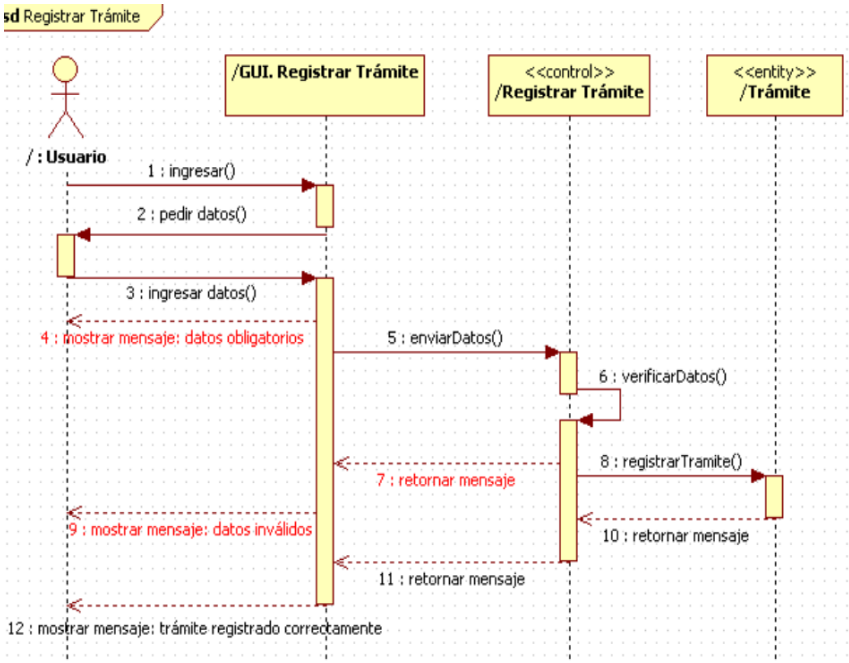


Fuente: Ramírez Coronado (2017)

3.3.1.3.3. Diagramas de secuencia

a) Registrar trámite

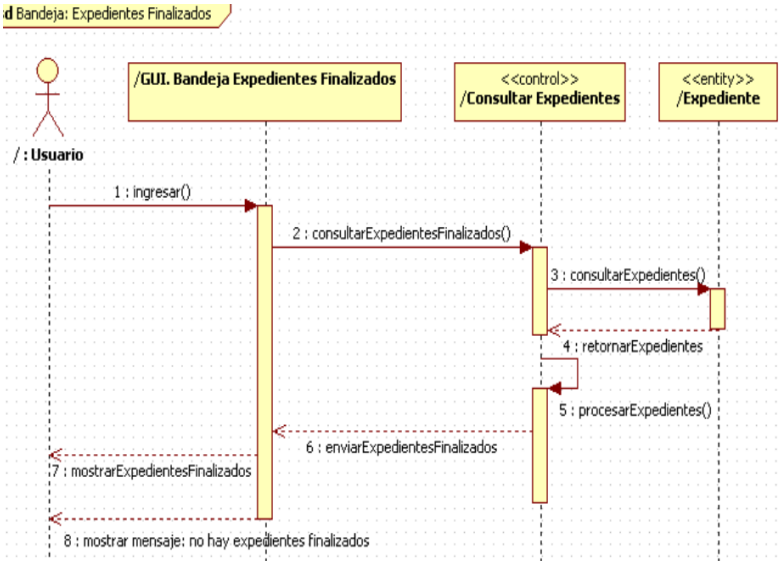
Ilustración 22. Diagrama de Secuencia - Registrar trámite



Fuente: Ramírez Coronado (2017)

b) Consultar bandeja expedientes finalizados

Ilustración 23. Diagrama de Secuencia - Bandeja expedientes finalizados

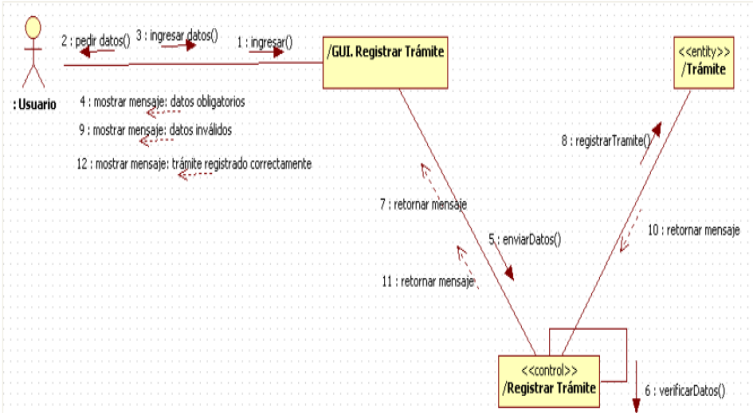


Fuente: Ramírez Coronado (2017)

3.3.1.3.4. Diagramas de colaboración

a) Registrar trámite

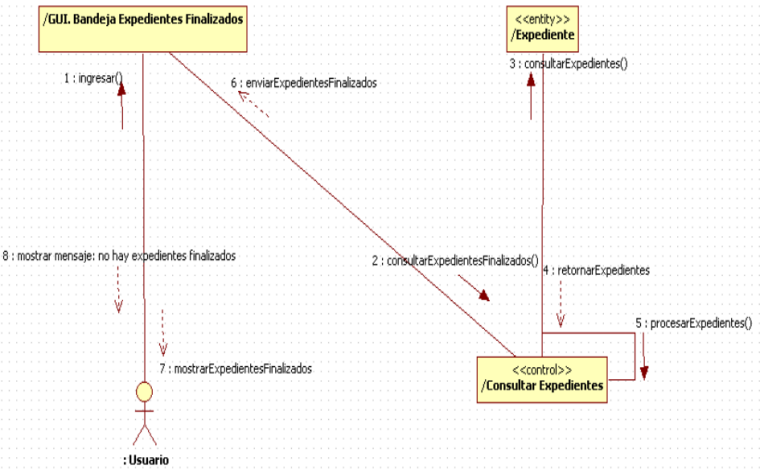
Ilustración 24. Diagrama de Colaboración - Registrar trámite



Fuente: Ramírez Coronado (2017)

b) Consultar bandeja expedientes finalizados

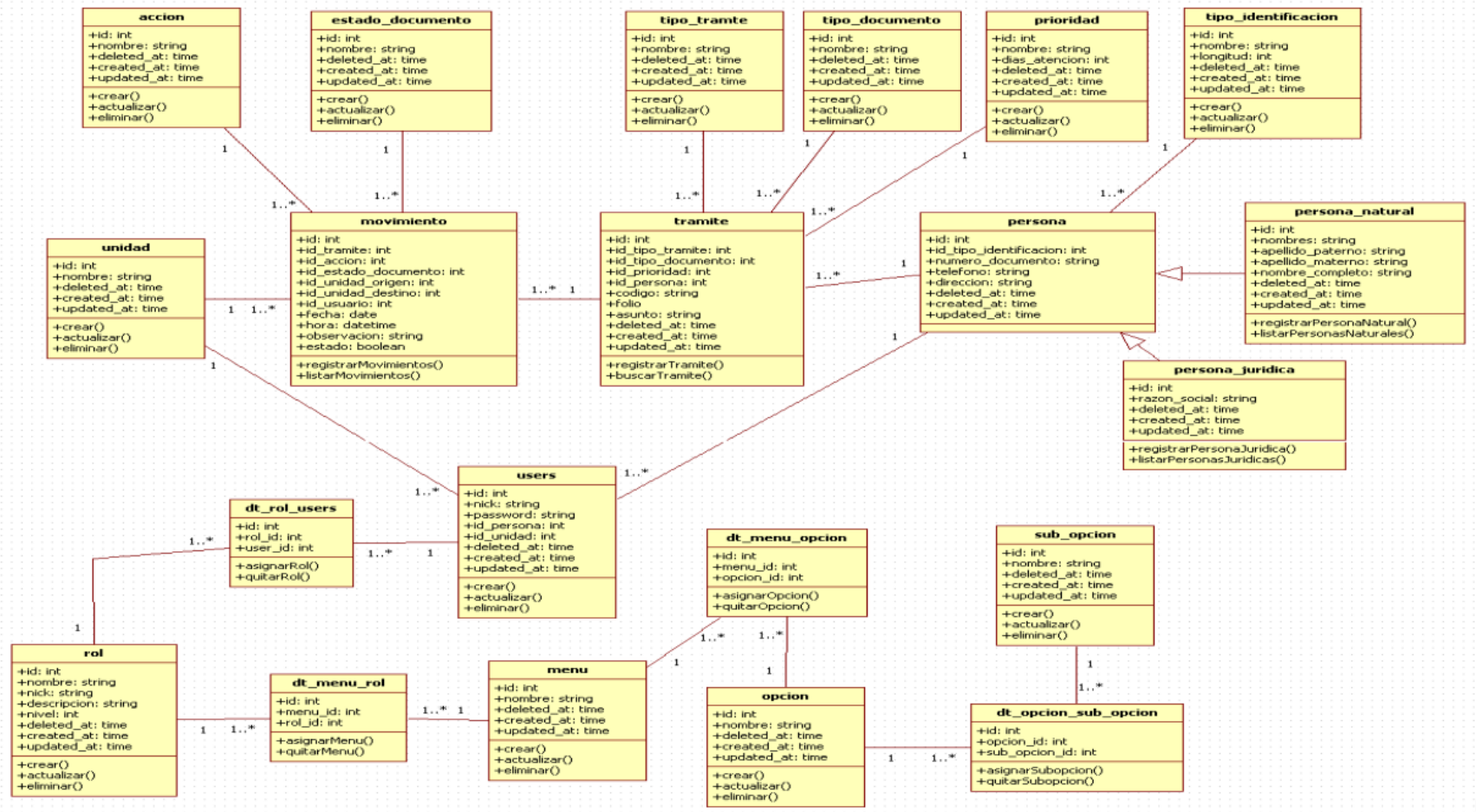
Ilustración 25. Diagrama de Colaboración - Consultar bandeja de expedientes finalizados



Fuente: Ramírez Coronado (2017)

3.3.1.3.5. Diagrama de Clases

Ilustración 26. Diagrama de clases - módulo tramite documentario

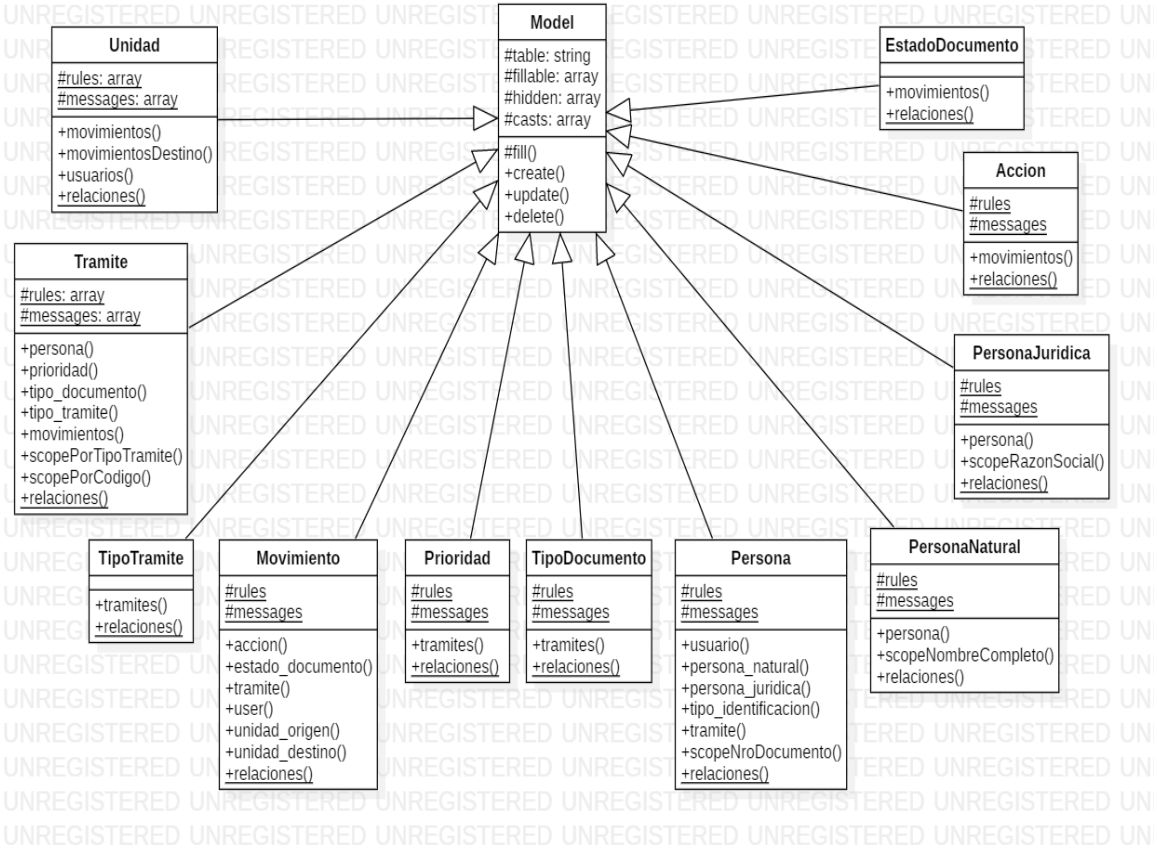


Fuente: Ramírez Coronado (2017)

3.3.1.3.6. Diagramas de Implementación en el framework Laravel

Diagrama de clases – Modelos

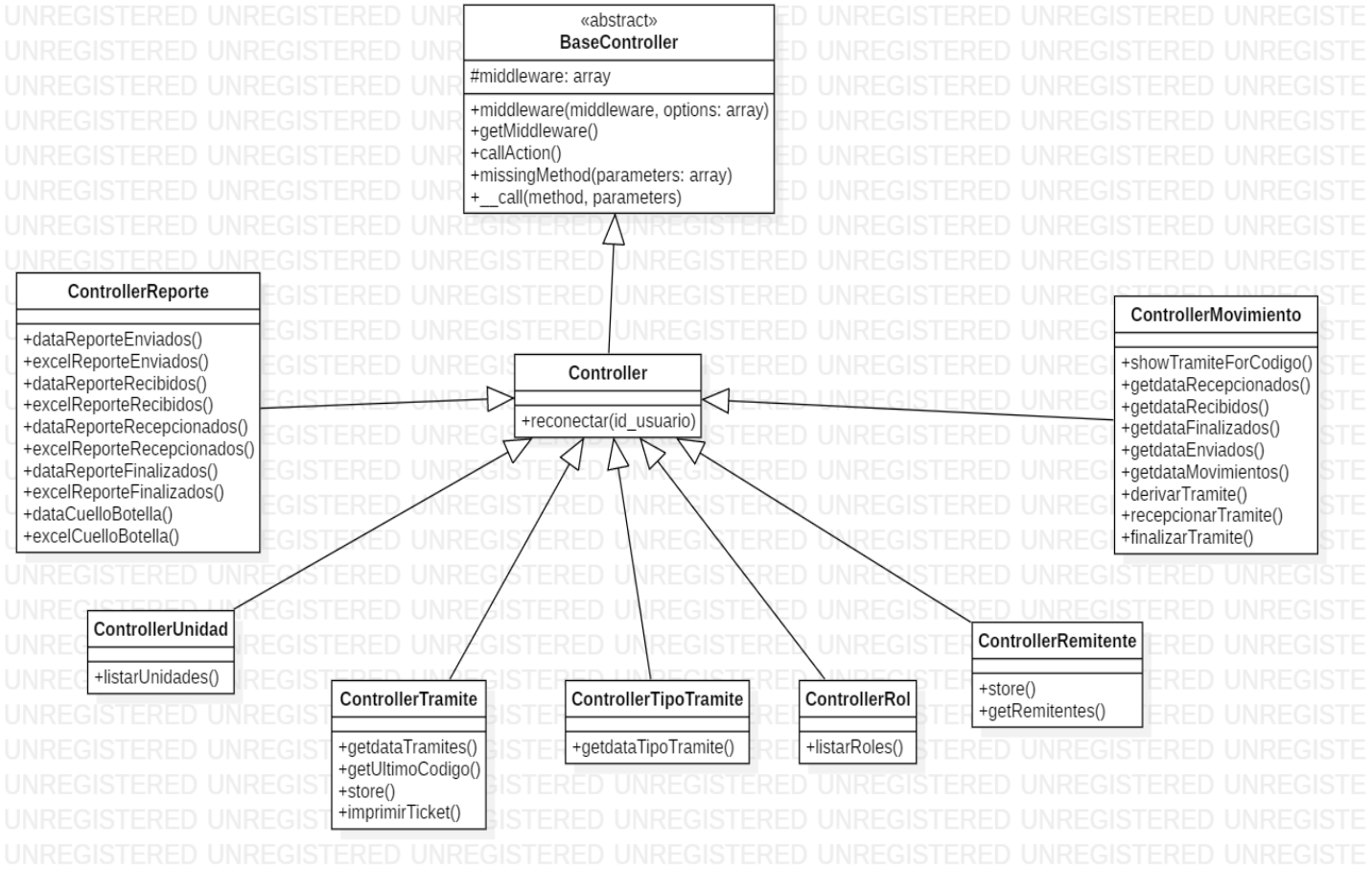
Ilustración 27. Diagrama de clases de implementación Laravel - Modelos



Fuente: Elaboración Propia

Diagrama de clases – Controladores

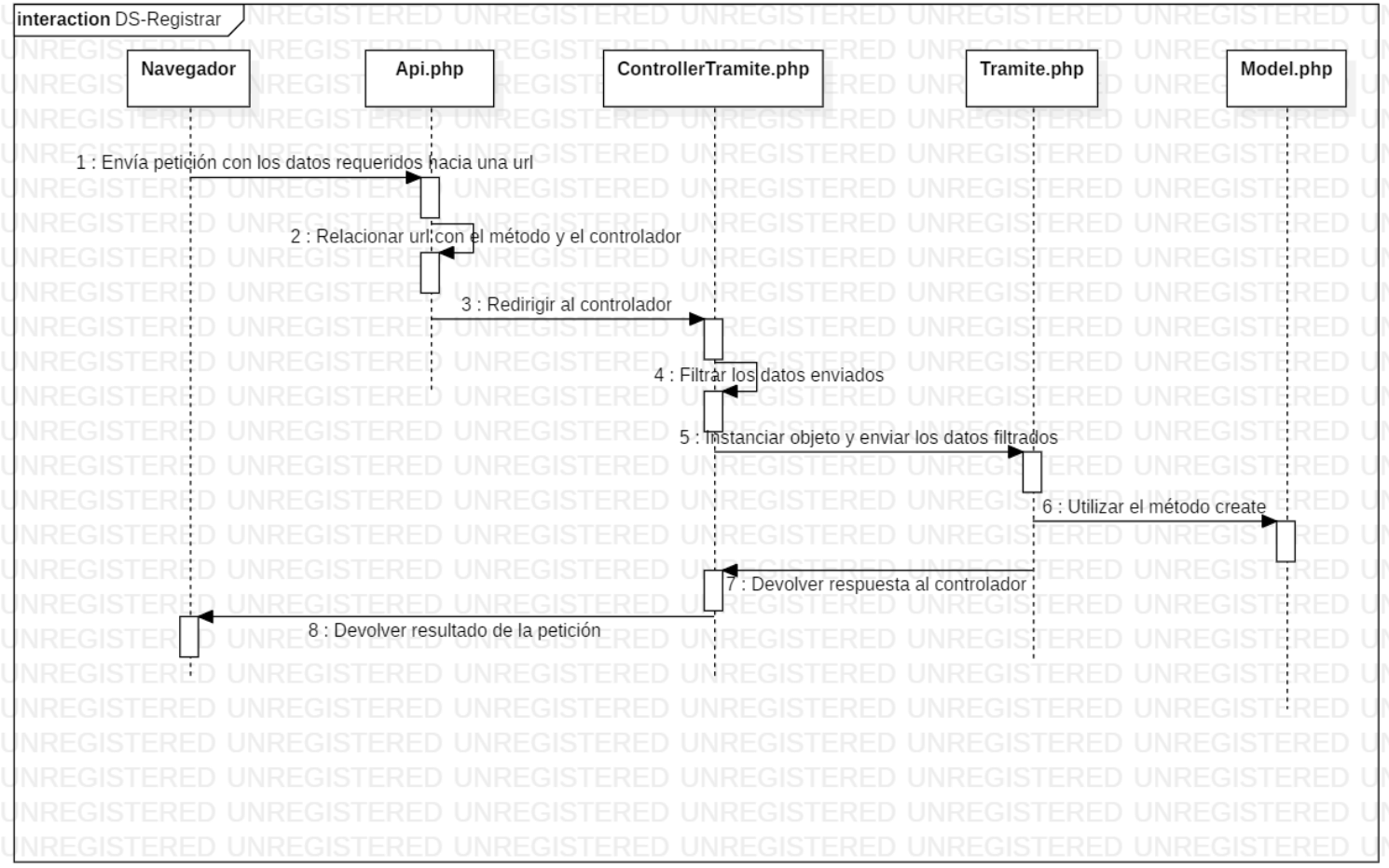
Ilustración 28. Diagrama de clases de implementación Laravel - Controladores



Fuente: Elaboración Propia

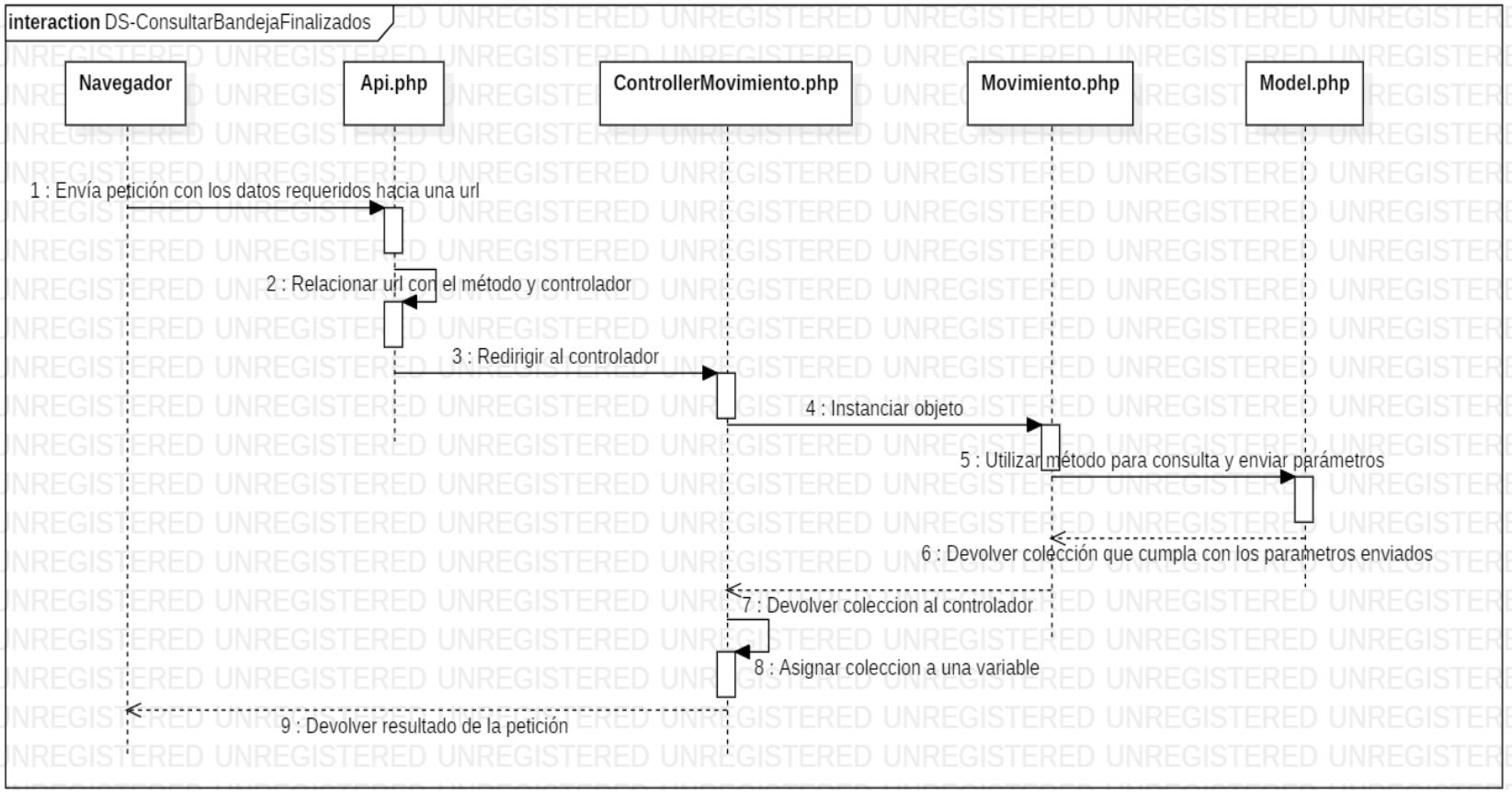
Diagramas de secuencia

Ilustración 29. Diagrama de secuencia de Implementación Laravel - registro de trámite



Fuente: Elaboración Propia

Ilustración 30. Diagrama de secuencia de Implementación Laravel - consultar bandeja de trámites finalizados

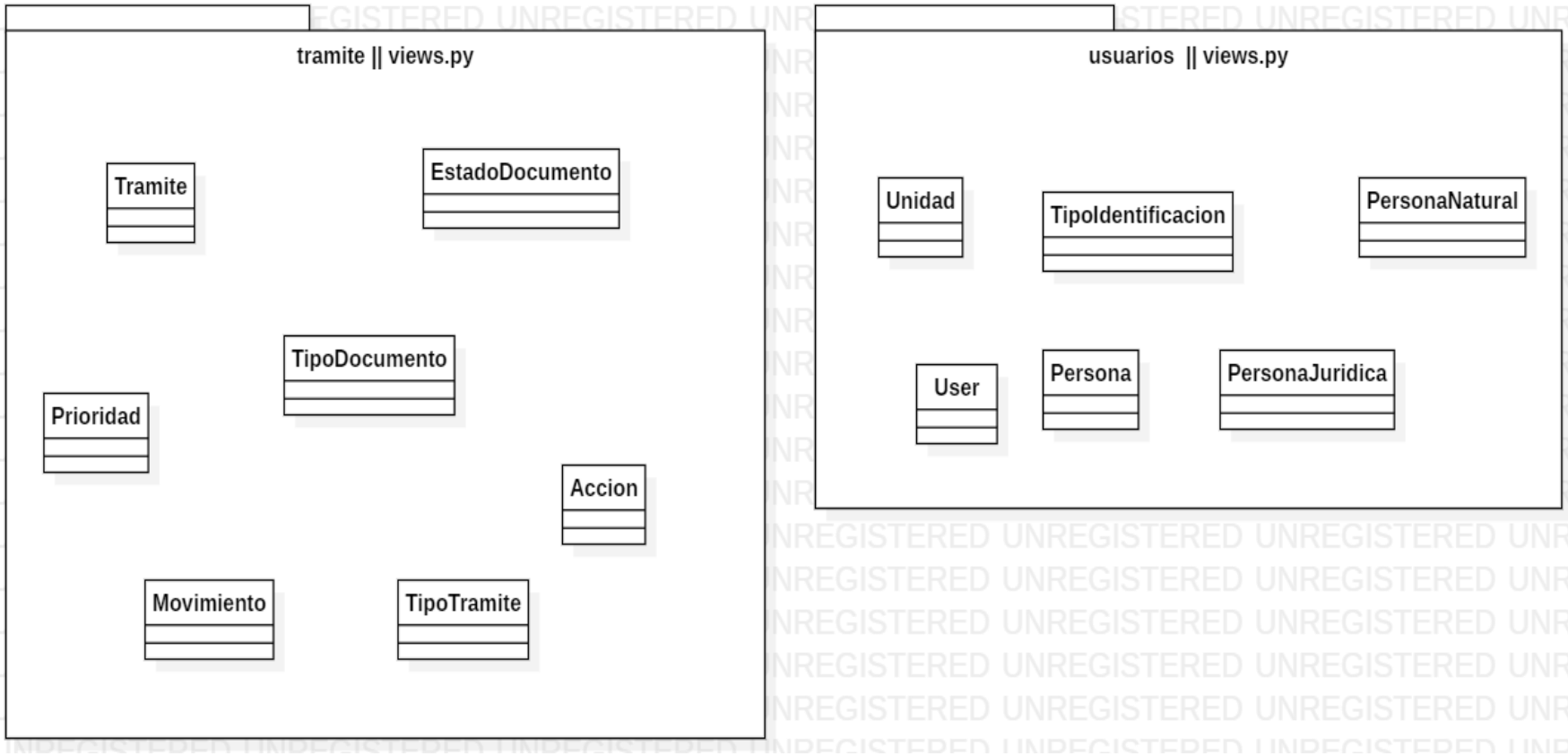


Fuente: Elaboración Propia

3.3.1.3.7. Diagramas de Implementación en el framework Django

Diagrama de clases – Modelos

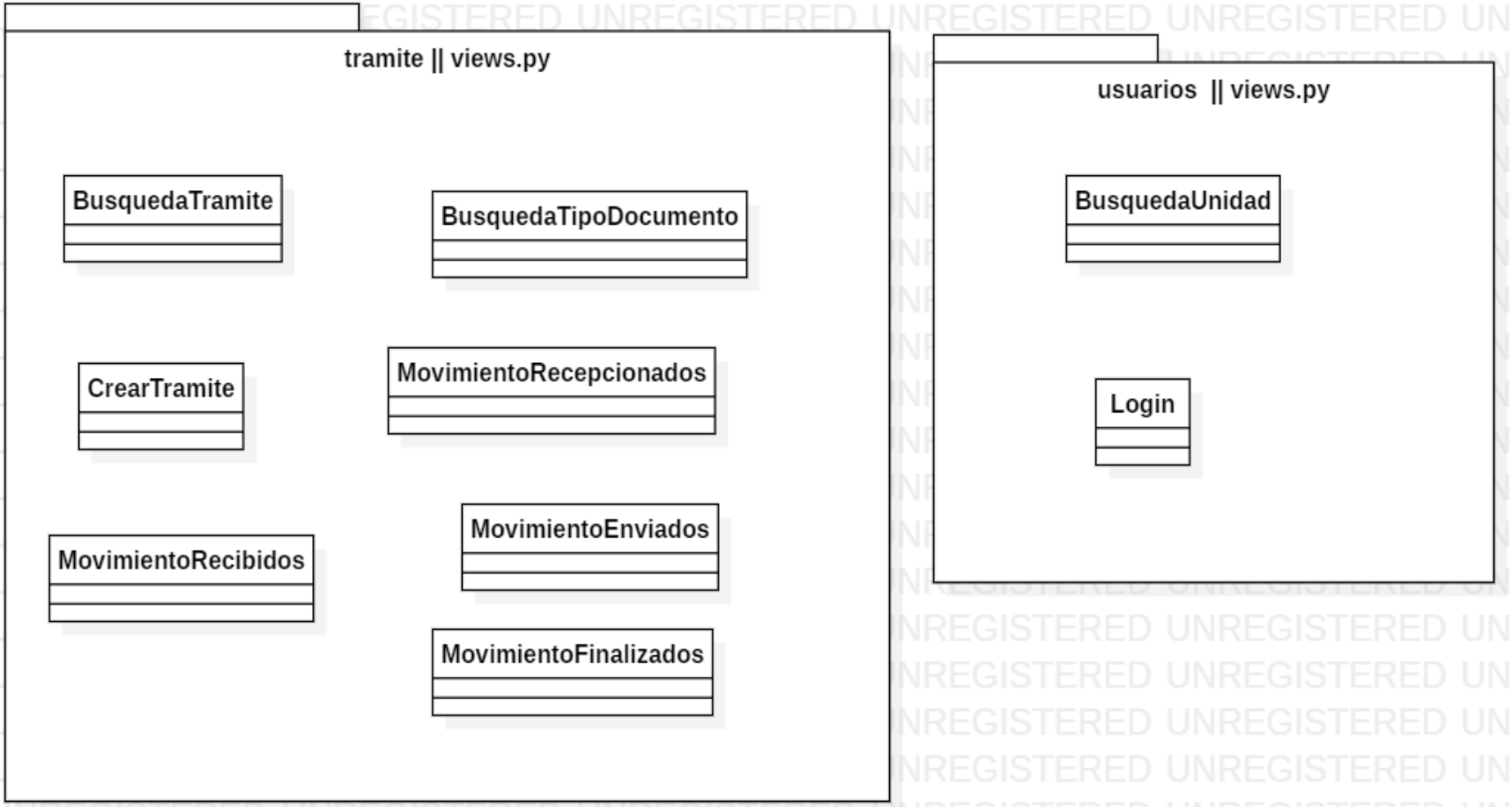
Ilustración 31. Diagrama de clases de implementación Django - Modelos



Fuente: Elaboración Propia

Diagrama de clases – Controladores

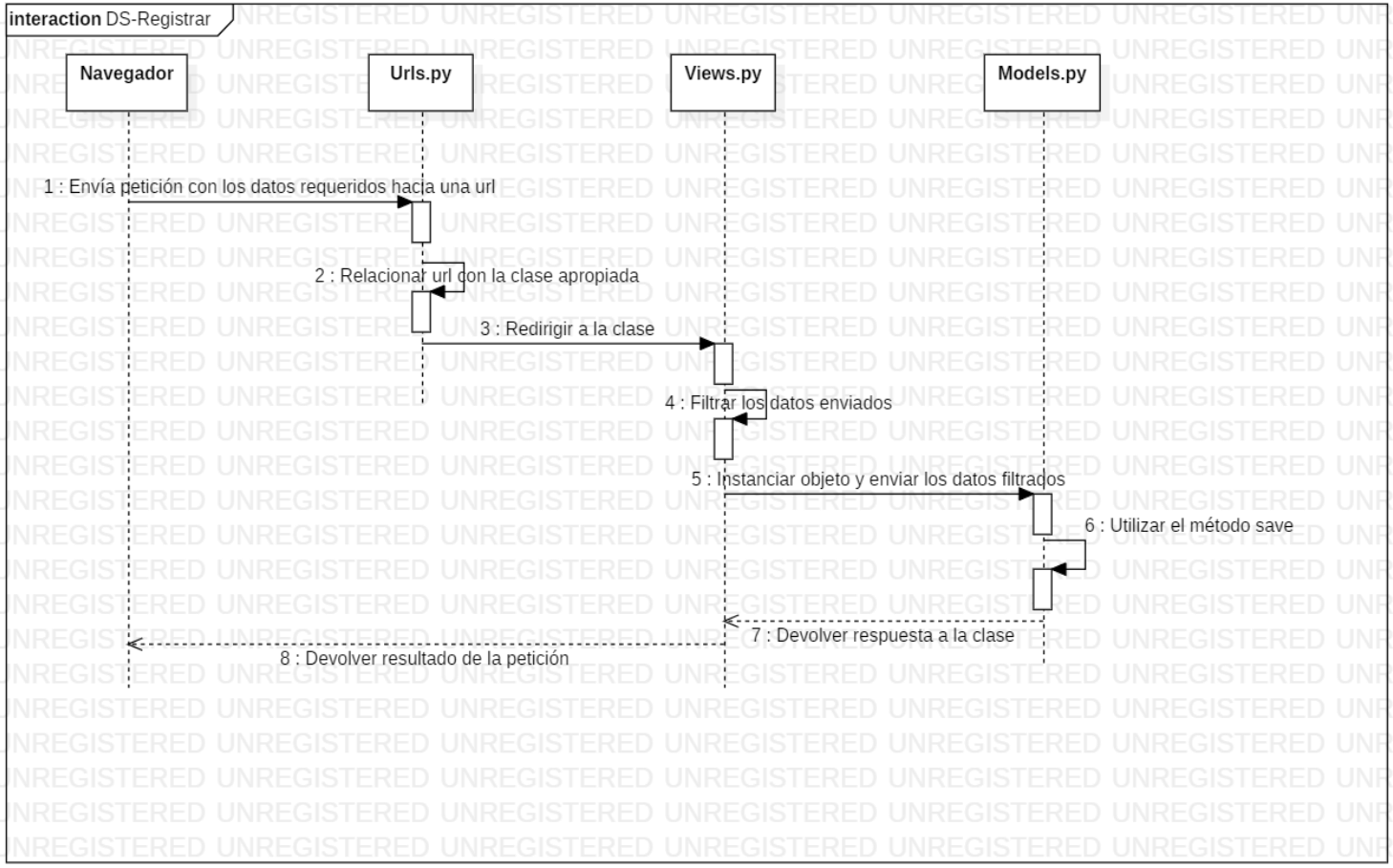
Ilustración 32. Diagrama de clases de implementación Django - Controladores



Fuente: Elaboración Propia

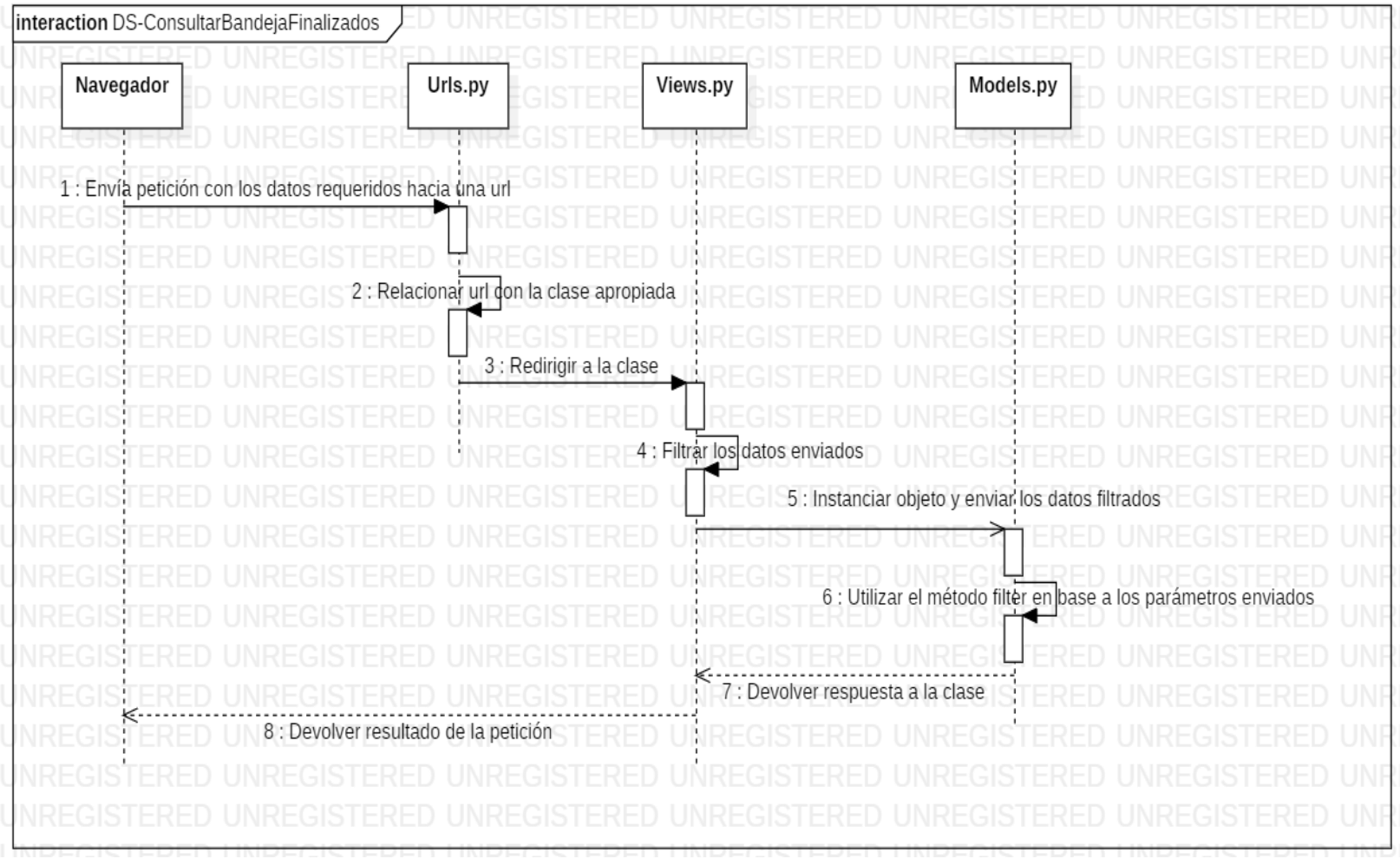
Diagramas de Secuencia

Ilustración 33. Diagrama de secuencia de Implementación Django - registro de trámite



Fuente: Elaboración Propia

Ilustración 34. Diagrama de secuencia de Implementación Django - consultar bandeja de trámites finalizados



Fuente: Elaboración Propia

3.3.2. Definición y ejecución de pruebas

Las pruebas fueron realizadas con el software Apache JMeter, el cual genera reportes de dos tipos: uno en donde se muestra el estado y respuesta de la petición y el otro muestra el tiempo de respuesta de cada petición, así como el promedio y desviación estándar de los tiempos.

Para llevar a cabo las pruebas, se estableció ciertos valores, necesarios para su correcta ejecución. El primer valor en ser fijado fue la cantidad de usuarios concurrentes, previamente en esta investigación se había definido tres valores: 1000, 2000 y 3000 usuarios.

El siguiente valor que se fijó fue el tiempo que le tomaría a los usuarios concurrentes para acceder a la ruta de operación, es decir, en cuantos segundos el total de usuarios deben haber ingresado a la ruta.

Por último, se configuró el valor correspondiente a la ruta o ubicación de la aplicación (de acuerdo al framework) así como puertos, en caso de no usar otros que no fueran los predeterminados.

Por otro lado, las especificaciones del software que fue utilizado en la realización de las pruebas se indican en la Tabla 8 y las especificaciones de las computadoras que fueron empleadas, todas del mismo modelo, se encuentran en la Tabla 9.

Tabla 8. Especificaciones de software

Tipo	Software
Sistema operativo	Windows 10
Motor de base de datos	MySQL 5.7 – 64 bits
Servidor de aplicaciones	Apache
Herramienta de testeo	Apache JMeter 5.0
Versión - PHP	7.2.11
Versión - Laravel	5.3
Versión Python	3.6.6
Versión Django	1.11

Elaboración propia

Tabla 9. Especificaciones de Hardware

Componente	Descripción
Procesador	Intel Core I7 2600 3.4 GHz
Disco duro	500 GB
Memoria RAM	8 GB

Elaboración propia

Cabe mencionar que previamente se realizaron pruebas en otros equipos físicos, que terminaron por descartarse pues no soportaban la carga de trabajo para los factores planteados. Las características de estos son mencionadas en el anexo 7.

Asimismo, también se pensó utilizar servicios externos como Amazon para poder alojar las aplicaciones, sin embargo, también se terminó descartando ya que no se encontró una forma de medir el uso de memoria RAM al ejecutarse una petición u operación (consulta o

registro) al menos remotamente; lo que si se podía con Apache JMeter con el tiempo de ejecución.

Junto con Apache JMeter (herramienta con la que se obtuvo los tiempos de respuesta para cada operación realizada) se utilizó JConsole, una herramienta de monitoreo que utiliza la máquina virtual de Java (Java VM) que proporciona información sobre el rendimiento y consumo de las aplicaciones. Con la ayuda de esta herramienta se pudo obtener información del uso de memoria RAM mientras se ejecutaba Apache JMeter.

Antes de realizar las pruebas también era necesario configurar el sistema operativo para que su rendimiento sea el óptimo cuando estas se lleven a cabo. Las tareas que se ejecutaron fueron las siguientes:

1. Configuración de programas que se inician automáticamente: para poder llevar a cabo esta configuración es necesario ir al Administrador de tarea del sistema operativo, luego dar clic en la pestaña Inicio, acto seguido se muestra una lista de programas y de acuerdo al criterio del usuario cada uno de ellos se irá deshabilitando. Debe tenerse en cuenta que se debe evitar programas que van asociados a componentes físicos del equipo (pantalla, sonido, red, etcétera).
2. Comprobar y reparar errores de disco duro: para examinar una unidad de disco duro se debe ingresar a la opción propiedades de la misma (normalmente este proceso se realiza con la unidad

donde se aloje el sistema operativo), posteriormente se selecciona la pestaña herramientas y en la sección comprobar errores se hará clic en el botón comprobar; luego desfragmentar el disco duro por cada prueba realizada; de las señalada en mención.

3. Eliminar manualmente archivos temporales: Existen carpetas del sistema operativo y usuario, que contienen archivos que puedes borrar periódicamente. Son 2 carpetas en total: Carpeta temporal de Windows, Carpeta temporal del usuario realizar este proceso con cada uno de los usuarios configurados en el equipo).

Con los elementos configurados se procedió a ejecutar las pruebas teniendo en cuenta la combinación de niveles de los 3 factores definidos en el diseño de la investigación (frameworks, número de usuarios y tipo de operación) correspondiendo a un total de 72 muestras tomadas. Sin embargo, de ese grupo, se decidió eliminar dos muestras por cada combinación (correspondiente al valor más bajo y al más alto) quedando con un total de 48 muestras con las que se trabajó posteriormente.

CAPITULO IV

RESULTADOS Y DISCUSIÓN

4.1. Resultados

Para procesar estadísticamente los datos obtenidos del tiempo de respuesta y memoria RAM se utilizó el software SPSS 25 utilizando la opción “Analizar modelo lineal general univariado”. Para ello es necesario organizar los datos siguiendo cierta estructura, para esta investigación y teniendo en cuenta los niveles y factores definidos se obtuvo la estructura similar a la mostrada en la tabla 15.

Tabla 10. Ejemplo de estructura en SPSS

nro_usuarios	Framework	Operación	Tiempo_ejecucion	Uso_memoria_ram
1000 usuarios	Django	registro	Valor1	Valor1
1000 usuarios	Django	consulta	Valor2	Valor2
2000 usuarios	Django	registro	Valor3	Valor3
2000 usuarios	Django	consulta	Valor4	Valor4
3000 usuarios	Django	registro	Valor5	Valor5
3000 usuarios	Django	consulta	Valor6	Valor6
1000 usuarios	Laravel	registro	Valor7	Valor7
1000 usuarios	Laravel	consulta	Valor8	Valor8
2000 usuarios	Laravel	registro	Valor9	Valor9
2000 usuarios	Laravel	consulta	Valor10	Valor10
3000 usuarios	Laravel	registro	Valor11	Valor11
3000 usuarios	Laravel	consulta	Valor12	Valor12

Elaboración propia

4.1.1. Tiempo de ejecución

En la tabla 16 se muestra el total de datos y los niveles de cada factor que fueron utilizados en las pruebas estadísticas.

Tabla 11. Factores y datos

Factores		Etiqueta de valor	N
nro_usuarios	1	1000 usuarios	16
	2	2000 usuarios	16
	3	3000 usuarios	16
framework	1	Django	24
	2	Laravel	24
operacion	1	registro	24
	2	consulta	24

Cabe indicar que ambas pruebas se realizaron empleando un nivel de significancia del 0.05, es decir, un intervalo de confianza del 95%, esperando obtener como error máximo posible un 5%.

En la tabla 17 se utiliza la columna Sig. para poder determinar si los factores o la interacción de estos presentan influencia significativa en la variable dependiente tiempo de respuesta. Si el valor es menor al nivel de significancia significa que el factor o la interacción es significativo sobre el tiempo de ejecución.

Tabla 12. Pruebas de efectos inter-sujetos – variable dependiente: tiempo de ejecución

Origen	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.
Modelo corregido	148130,417[a]	11	13466,402	209,911	,000
Intersección	671660,083	1	671660,083	10469,696	,000
nro_usuarios	353,167	2	176,583	2,753	,077
framework	36741,333	1	36741,333	572,716	,000
operacion	88408,333	1	88408,333	1378,090	,000
nro_usuarios * framework	368,667	2	184,333	2,873	,070
nro_usuarios * operacion	146,167	2	73,083	1,139	,331
framework * operacion	21590,083	1	21590,083	336,542	,000
nro_usuarios * framework * operacion	522,667	2	261,333	4,074	,025
Error	2309,500	36	64,153		
Total	822100,000	48			
Total corregido	150439,917	47			
a. R al cuadrado = ,985 (R al cuadrado ajustada = ,980)					

Los factores e interacciones que cumplen con la condición mencionada anteriormente son:

- framework
- operación
- framework * operación
- nro_usuarios * framework * operacion

Todos estos factores e interacciones influyen significativamente sobre la variable independiente tiempo de ejecución.

Al realizar la prueba estadística se puede aplicar una configuración que permite generar un apartado en los resultados de la prueba llamado medias marginales. Para esta prueba en particular se obtuvieron los resultados mostrados en las tablas 19, 20 que permite comprobar la diferencia existente por factor.

Tabla 13. Media marginal factor: framework. Variable dependiente: tiempo_ejecucion

framework	Media	Desv. Error	Intervalo de confianza al 95%	
			Límite inferior	Límite superior
Django	90,625	1,635	87,309	93,941
Laravel	145,958	1,635	142,643	149,274

Tabla 14. Media marginal factor: operacion. Variable dependiente: tiempo_ejecucion

operacion	Media	Desv. Error	Intervalo de confianza al 95%	
			Límite inferior	Límite superior
registro	161,208	1,635	157,893	164,524
consulta	75,375	1,635	72,059	78,691

De manera específica se puede decir que la media marginal estimada para el factor framework respecto a la variable dependiente tiempo de ejecución varía por nivel. En este caso la media estadística para el framework Django (90.625 ms) es menor respecto a la del framework Laravel (145.958 ms).

También se observa el mismo comportamiento para el factor operación. El tiempo de ejecucion de la operación de consulta (75.375 ms) es menor al de la operación de registro (161.208 ms).

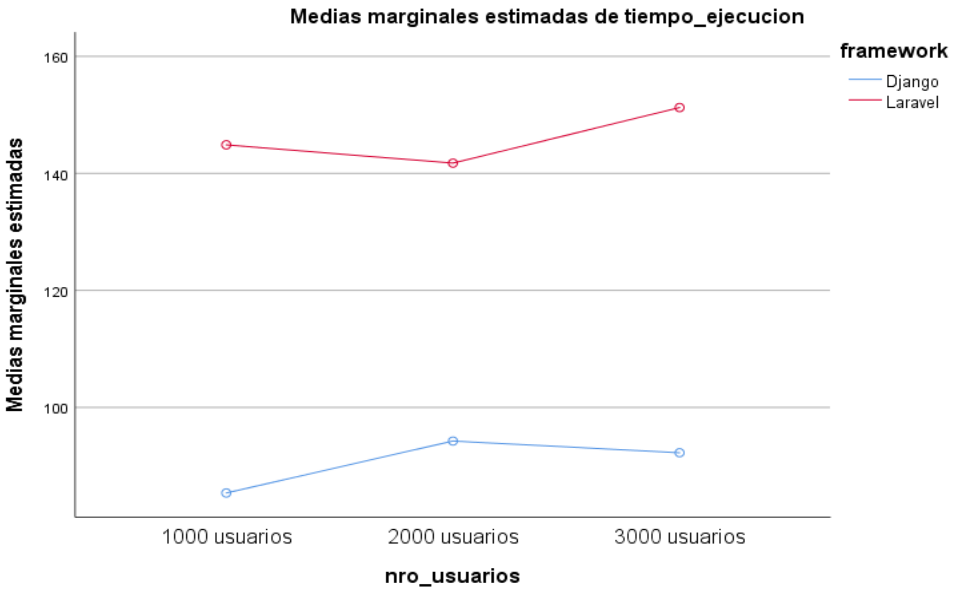
De acuerdo a las configuraciones realizadas en la prueba, el software SPSS, genera también gráficos por cada interacción. Al igual que el caso anterior se agregan los correspondientes a la interacción de nro_usuarios * framework (Gráfico 1 y Tabla 21) y nro_usuarios * operación (Gráfico 2 y Tabla 22).

Tabla 15. Medias marginales estimadas tiempo de ejecución - nrousuario y framework

nro_usuarios	framework	Media	Desv. Error	Intervalo de confianza al 95%	
				Lím. inferior	Lím. superior
1000 usuarios	Django	85,375	2,832	79,632	91,118
	Laravel	144,875	2,832	139,132	150,618
2000 usuarios	Django	94,250	2,832	88,507	99,993
	Laravel	141,750	2,832	136,007	147,493
3000 usuarios	Django	92,250	2,832	86,507	97,993
	Laravel	151,250	2,832	145,507	156,993

Fuente: Obtenido a partir del software SPSS

Gráfico 1. Medias marginales estimadas tiempo de ejecución - nrousuario y framework



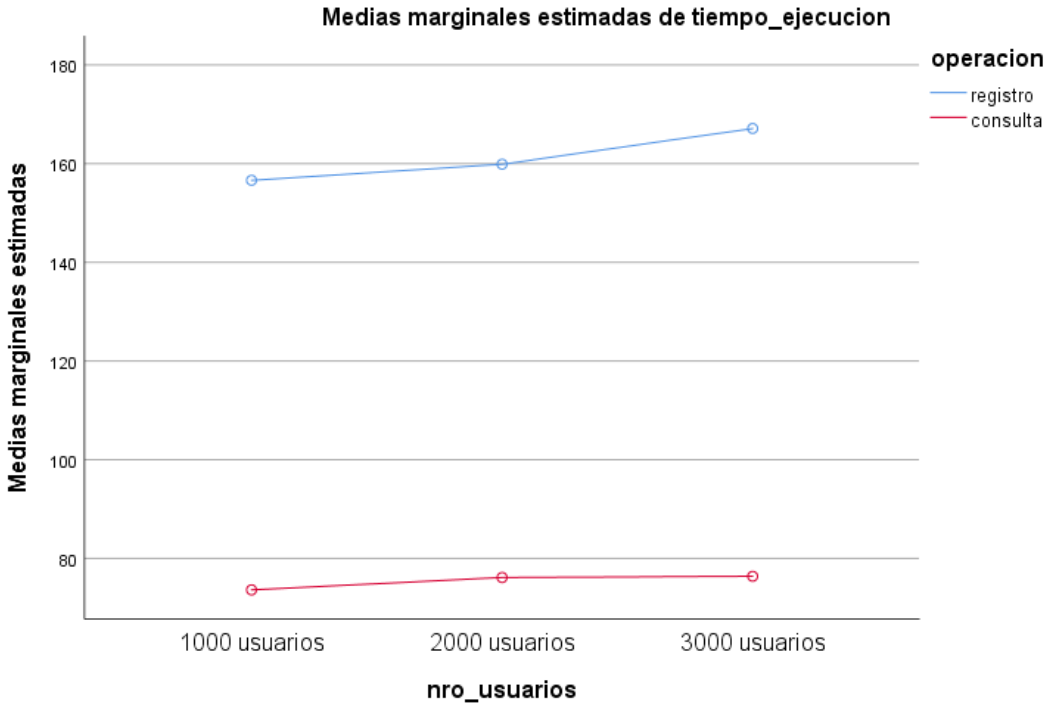
Fuente: Obtenido a partir del software SPSS

Tabla 16. Medias marginales estimadas tiempo de ejecucion - nrousuarios y operación

nro_usuarios	operacion	Media	Desv. Error	Intervalo de confianza al 95%	
				Lím. inferior	Lím. superior
1000 usuarios	registro	156,625	2,832	150,882	162,368
	consulta	73,625	2,832	67,882	79,368
2000 usuarios	registro	159,875	2,832	154,132	165,618
	consulta	76,125	2,832	70,382	81,868
3000 usuarios	registro	167,125	2,832	161,382	172,868
	consulta	76,375	2,832	70,632	82,118

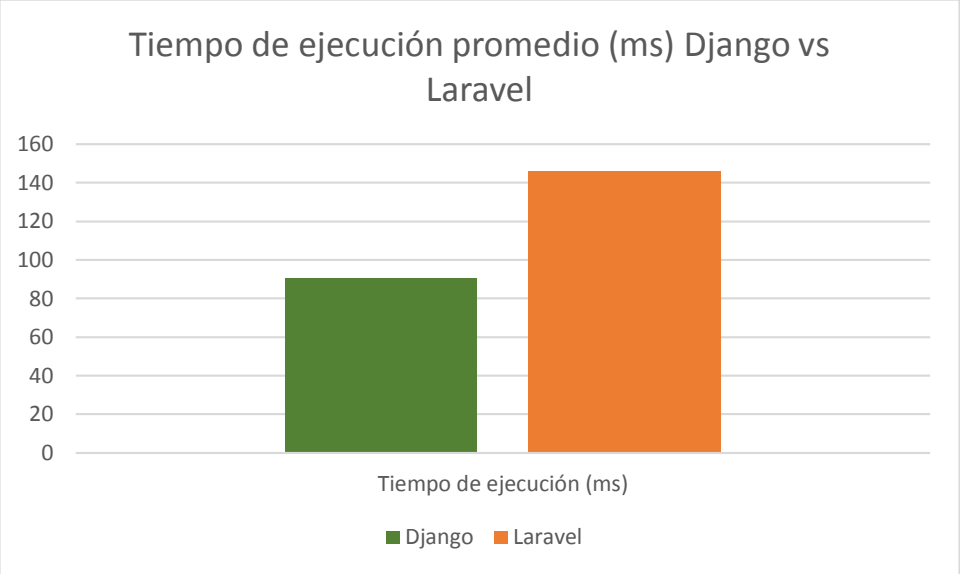
Fuente: Obtenido a partir del software SPSS

Gráfico 2. Medias marginales estimadas tiempo de ejecucion - nrousuarios y operación



Fuente: Obtenido a partir del software SPSS

Gráfico 3. Gráfico de barras Tiempo de ejecución promedio (ms) Django vs Laravel



Fuente: Elaboración propia

4.1.2. Uso de memoria RAM

Para analizar los datos se tuvieron en cuenta las mismas consideraciones que en el caso anterior, obteniendo los siguientes resultados:

Tabla 17. Pruebas de efectos inter-sujetos – variable dependiente: uso de memoria RAM

Origen	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.
Modelo corregido	1158203,178[a]	11	105291,198	15,231	,000
Intersección	3059866,725	1	3059866,725	442,624	,000
nro_usuarios	667870,170	2	333935,085	48,305	,000
framework	48047,871	1	48047,871	6,950	,012
operacion	107575,605	1	107575,605	15,561	,000
nro_usuarios * framework	53463,725	2	26731,863	3,867	,030

nro_usuarios * operacion	215011,242	2	107505,621	15,551	,000
framework * operacion	8864,757	1	8864,757	1,282	,265
nro_usuarios * framework * operacion	57369,807	2	28684,904	4,149	,024
Error	248868,770	36	6913,021		
Total	4466938,674	48			
Total corregido	1407071,948	47			
a. R al cuadrado = ,823 (R al cuadrado ajustada = ,769)					

Al igual que el caso anterior se debe identificar los factores e interacciones que cumplen con la condición de ser menor al nivel de significancia utilizada (0.05). En este caso son los siguientes:

- nro_usuarios
- framework
- operación
- nro_usuarios * framework
- nro_usuarios * operación
- nro_usuarios * framework * operación

Para poder identificar que promedio de uso de memoria RAM respecto al factor framework, la prueba también se configuró para obtener las medias marginales, obteniéndose los datos mostrados en las tablas 22 y 23, correspondientes a los factores framework y operación.

Tabla 18. Media marginal factor: framework. Variable dependiente: uso_memoria_ram

framework	Media	Desv. Error	Intervalo de confianza al 95%	
			Límite inferior	Límite superior
Django	284,121	16,972	249,700	318,541
Laravel	220,844	16,972	186,423	255,264

Tabla 19. Media marginal factor: operacion. Variable dependiente: uso_memoria_ram

operacion	Media	Desv. Error	Intervalo de confianza al 95%	
			Límite inferior	Límite superior
registro	299,823	16,972	265,403	334,243
consulta	205,141	16,972	170,721	239,562

Interpretando ambas tablas se puede decir que la media marginal estimada para el factor framework respecto a la variable dependiente uso de memoria RAM varía entre ambos niveles, es decir la media estadística de Django (284.121 MB) es mayor respecto a la de Laravel (220,844 MB).

El uso de memoria RAM también varía respecto al factor operación, siendo la media marginal de consulta (205,141 MB) menor en comparación a la de registro (299,823 ms).

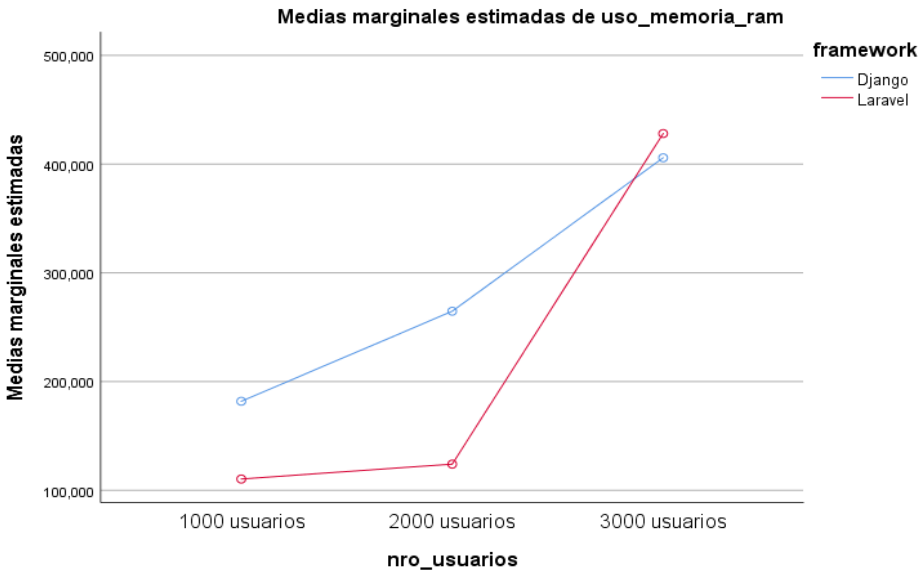
Los gráficos 3 y 4, generados por cada factor permiten comprobar los datos mostrados en las tablas.

Tabla 20. Medias marginales uso de memoria ram - nro usuarios y framework

nro_usuarios	framework	Media	Desv. Error	Intervalo de confianza al 95%	
				Lím. inferior	Lím. superior
1000 usuarios	Django	181,845	29,396	122,228	241,463
	Laravel	110,392	29,396	50,774	170,010
2000 usuarios	Django	264,686	29,396	205,069	324,304
	Laravel	124,055	29,396	64,437	183,673
3000 usuarios	Django	405,830	29,396	346,212	465,448
	Laravel	428,083	29,396	368,465	487,701

Fuente: Obtenido a partir del software SPSS

Gráfico 4. Medias marginales uso de memoria ram - nro usuarios y framework



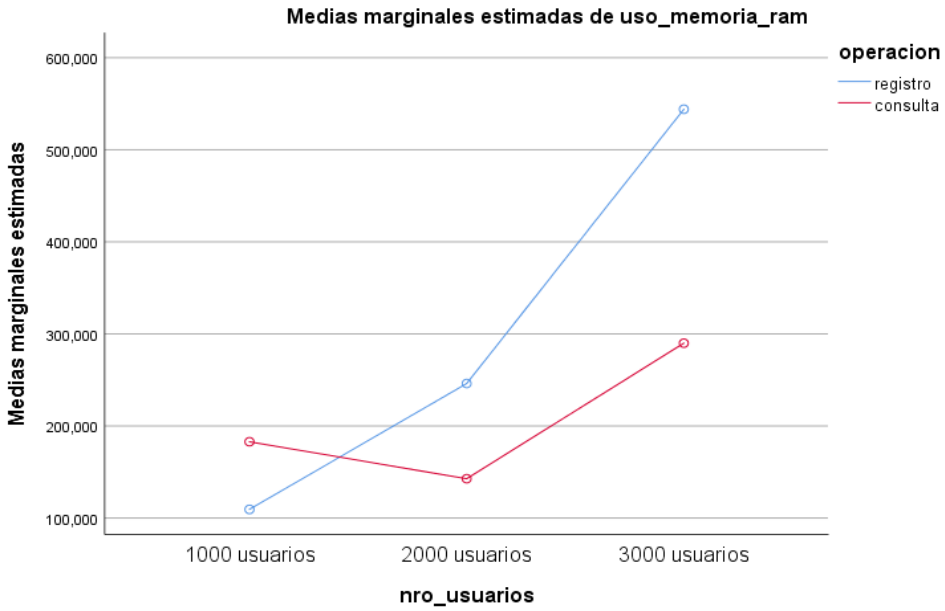
Fuente: Obtenido a partir del software SPSS

Tabla 21. Medias marginales uso de memoria ram – nrousuarios y operacion

nro_usuarios	operacion	Media	Desv. Error	Intervalo de confianza al 95%	
				Lím. inferior	Lím. superior
1000 usuarios	registro	109,390	29,396	49,772	169,008
	consulta	182,847	29,396	123,230	242,465
2000 usuarios	registro	246,085	29,396	186,468	305,703
	consulta	142,656	29,396	83,038	202,274
3000 usuarios	registro	543,993	29,396	484,375	603,611
	consulta	289,920	29,396	230,302	349,538

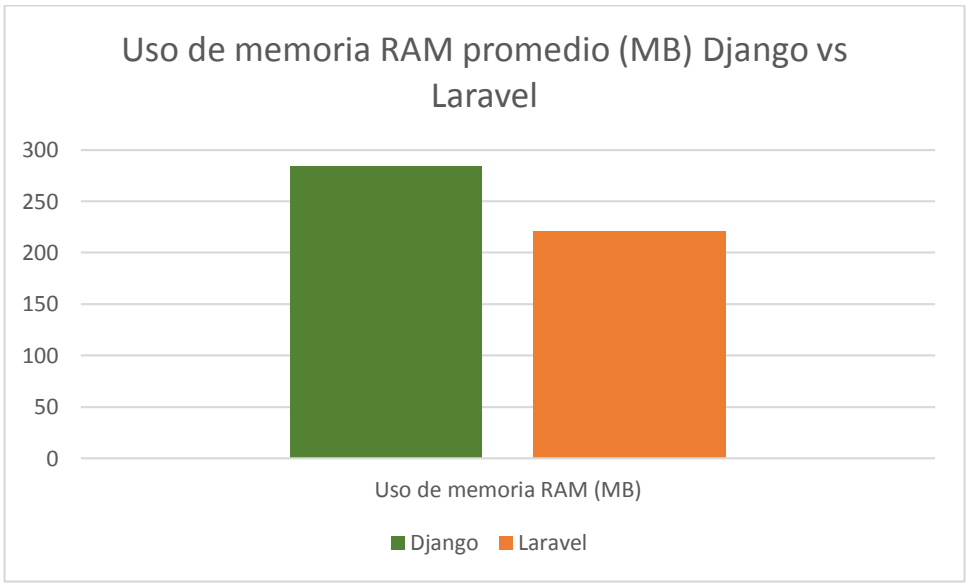
Fuente: Obtenido a partir del software SPSS

Gráfico 5. Medias marginales uso de memoria ram – nrousuarios y operacion



Fuente: Obtenido a partir del software SPSS

Gráfico 6. Gráfico de barras Uso de memoria RAM promedio (MB) Django vs Laravel



Fuente: Elaboración propia

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Con los resultados de las pruebas estadísticas realizadas se puede afirmar que el rendimiento, abarcado por las variables dependientes tiempo de ejecución y uso de memoria RAM, se ve influenciado de manera significativa por los factores *framework* y t*ipo de operación*, ya que las medias marginales varían para cada variable. Esto lleva a aceptar la hipótesis alternativa general, es decir que existe una variación en el rendimiento de las aplicaciones desarrolladas en Laravel y Django.

- Los marcos de trabajo tienen su propia estructura de trabajo y configuración para cada proyecto creado.

- Para este estudio en particular si existe una variación de la variable dependiente tiempo de ejecución, esto se puede confirmar a través de la prueba estadística de análisis lineal general univariado, que indica no solamente que el *framework* es un factor significativo, sino que, además, a través de las medias marginales se puede interpretar que el tiempo de ejecucion para Django (90.625 ms) es menor al de Laravel (145.958 ms), aceptando la hipótesis especifica 1.

- Para el caso de la hipótesis específica 2 la situación se invierte respecto a la anterior, es decir, existe una variación de la variable dependiente uso de memoria RAM. Sin embargo, los resultados de las medias marginales indican que dicha el valor de esta variable para Laravel (220,844 MB) es menor respecto al de Django (284.121 MB).

5.2. Recomendaciones

- Al realizar las pruebas para 3000 usuarios se observó que en algunas muestras la petición devolvía un estado de error, cabe aclarar que en un margen menor al 10%. Por ello se recomienda tomar un rango de usuarios menor que al tomado en esta investigación.

- Se recomienda leer la documentación oficial de cada framework, para poder explotar al máximo las cualidades de cada framework.

- Las unidades de medida utilizadas para las variables tiempo de ejecución y uso de memoria RAM fueron milisegundos y megabytes (MB) respectivamente. Se recomienda continuar la investigación en otras unidades de medida (décimas de segundo o gigabytes, por ejemplo), debido a que se presume existe una influencia en el tiempo debido a los servicios del sistema operativo. O en su defecto realizar estudios posteriores que tengan la presencia de diversos sistemas operativos.

- Utilizar otros sistemas operativos para realizar futuras comparaciones, en este proyecto se empleó Windows 10 debido a la facilidad y rapidez para poner en funcionamiento las herramientas empleadas, sin embargo, se invita a realizar futuras comparaciones en sistemas basados en Linux.

- Realizar alianzas públicas - privadas; para así orientar recursos a la creación de potentes laboratorios con la instalación y puesta en funcionamiento de cluster de alta gama de servidores orientado a la investigación y poder realizar este tipo de estudios mucho más detallado y lograr resultados de calidad, y asi conseguir información refinada de las altas tecnologías como los nuevos frameworks, otros

estudios como la inteligencia artificial, etc; para la creación de software de calidad a nivel de cliente- servidor; está recomendación se hace en el marco, que en nuestra región y por ende a nivel nacional, no se cuenta con este tipo de servicios, y la generación de nuevo software se respalda mayormente en la documentación generada por el fabricante, y esta orientado a otra realidad geográfica; y no en la praxis o comprobandose en el campo de los hechos; lo cual trae problemas comunes en los usuarios finales como la lentitud en los tiempos de respuesta, o el uso elevado de recursos puesta en producción por no realizarse estas pruebas; esta investigación se realizo con recursos propios resultandome costoso asumirlo.

BIBLIOGRAFÍA

Bakieva, M., Gonzáles Such, J., & Jornet, J. . (2015). SPSS: ANOVA de un factor. España.

Obtenido de https://www.uv.es/innomide/spss/SPSS/SPSS_0702b.pdf

Bean, M. (2015). Laravel 5 Essentials. Inglaterra: Packt Publishing. Obtenido de http://file.allitebooks.com/20150731/Laravel%205%20Essentials.pdf

Calmet Izquierdo, J. P. (2014). Sistema informático web de trámite documentario para la Ugel de Zarumilla – Tumbes utilizando los frameworks AngularJS Y Spring MVC. Trujillo, Perú. Obtenido de http://repositorio.upao.edu.pe/handle/upaorep/642

DELL. (1 de Junio de 2018). *De qué manera la memoria de acceso aleatorio (RAM) afecta el rendimiento*. Obtenido de https://www.dell.com/support/article/pe/es/pedhs1/sln179266/de-qué-manera-la-memoria-de-acceso-aleatorio-ram-afecta-el-rendimiento?lang=es

Django Project. (s.f.). *Django Project*. Obtenido de Django Project: https://docs.djangoproject.com/es/2.0/intro/overview/

Enrech Enrech, S. (29 de Enero de 2013). Diseño de una aplicación Web para el control de los cultivos frutales. España. Obtenido de https://repositori.udl.cat/bitstream/handle/10459.1/46610/senreche.pdf?sequence=1

Everts, T. (16 de Noviembre de 2015). *A Brief History of Web Performance ROI*. Obtenido de https://dzone.com/articles/a-brief-history-of-Web-performance-roi

GNUSTEP. (s.f.). *¿Que es un Framework?* Obtenido de ¿Que es un Framework?: https://gnustep.wordpress.com/gnustep-a-fondo/%C2%BFque-es-un-framework-%C2%BFcomo-se-utiliza/

Google Trends. (2017). *trends.google.es*. Obtenido de https://trends.google.es/trends/explore?q=laravel,django

Guerrero Benalcázar, R. I. (2016). *Estudio comparativo de los frameworks Ruby on rails y Django para la implementación de un sistema informático de control y administración de network marketing.* Perú: Editorial Universidad Técnica del Norte.

Hernández Sampieri, R. (2010). *Metodología de la investigación.* México D.F.: McGraw-Hill.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. d. (2014). Metodología de la Investigación. (6). México: Mc Graw Hill. Obtenido de http://observatorio.epacartagena.gov.co/wp-content/uploads/2017/08/metodologia-de-la-investigacion-sexta-edicion.compressed.pdf

Holovaty, A., & Kaplan Moss, J. (28 de Julio de 2008). El libro de Django. Obtenido de http://www.interaktiv.cl/blog/wp-content/uploads/2013/10/django-book-es-1.0-0.1-r757.pdf

IBM. (1993). Dictionary of IBM & Computing Terminology. USA: McGraw-Hill. Obtenido de https://www.ibm.com/ibm/history/documents/pdf/glossary.pdf

JMeter, A. (2019). *Apache JMeter*. Obtenido de https://jmeter.apache.org

Laravel. (2019). Obtenido de https://laravel.com/docs/5.8#installation

Martín Díaz, O. (2006). Modelado de Negocio (Diagrama de Actividades). Sevilla, España. Obtenido de http://www.lsi.us.es/docencia/get.php?id=2007

McCool, S. (2012). Laravel Starter. (P. Publishing, Ed.) Birmingham, Inglaterra. Obtenido de http://blog.flds.fr/site/assets/files/1212/laravel_starter.pdf

MDN web docs. (18 de Marzo de 2019). *Framework Web Django (Python)*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Django

Microsoft Developer Network. (29 de Septiembre de 2016). *Generate and run a coded web performance test*. Obtenido de https://msdn.microsoft.com/es-pe/library/ms182552.aspx

MLG. ANOVA factorial univariante. (s.f.). El modelo lineal general. Obtenido de https://www.fibao.es/media/uploads/manual_spss_capítulos_12_hasta_el_final.pdf

Mozilla. (s.f.). *Developer Mozilla*. Obtenido de Developer Mozilla: https://developer.mozilla.org/

Rahm, E. (15 de Mayo de 2017). *Firefox memory usage with multiple content processes*. Obtenido de http://www.erahm.org/2017/05/15/firefox-memory-usage-with-multiple-content-processes/

Ramirez Coronado, L. G. (2017). *Análisis comparativo del tiempo de ejecución de una aplicación Web desarrollada en diferentes marcos de trabajo en el lado cliente y en el lado servidor.* Piura: Universidad Nacional de Piura.

Sierra, F., Acosta, J., Ariza, J., & Salas, M. (s.f). *Estudio y análisis de los framework en PHP basados en el modelo vista controlador para el desarrollo de software orientado a la web.* Perú.

SimilarTech. (6 de Junio de 2019). *SimilarTech*. Obtenido de https://www.similartech.com/compare/django-vs-laravel

Soasta. (2016). *2016 Holiday Retail Insights Report.*

Stack Overflow. (2016). *Stack Overflow Developer Survey 2016 Results*. Obtenido de https://insights.stackoverflow.com/survey/2016

The Tech Terms Computer Dictionary. (7 de Marzo de 2013). *https://techterms.com/definition/framework*. Obtenido de https://techterms.com/definition/framework

Villegas Avilés, J. E. (29 de Febrero de 2012). Análisis y desarrollo de un programa de lealtad aplicado a casinos. México. Obtenido de http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/226/A7.pdf?sequence=7

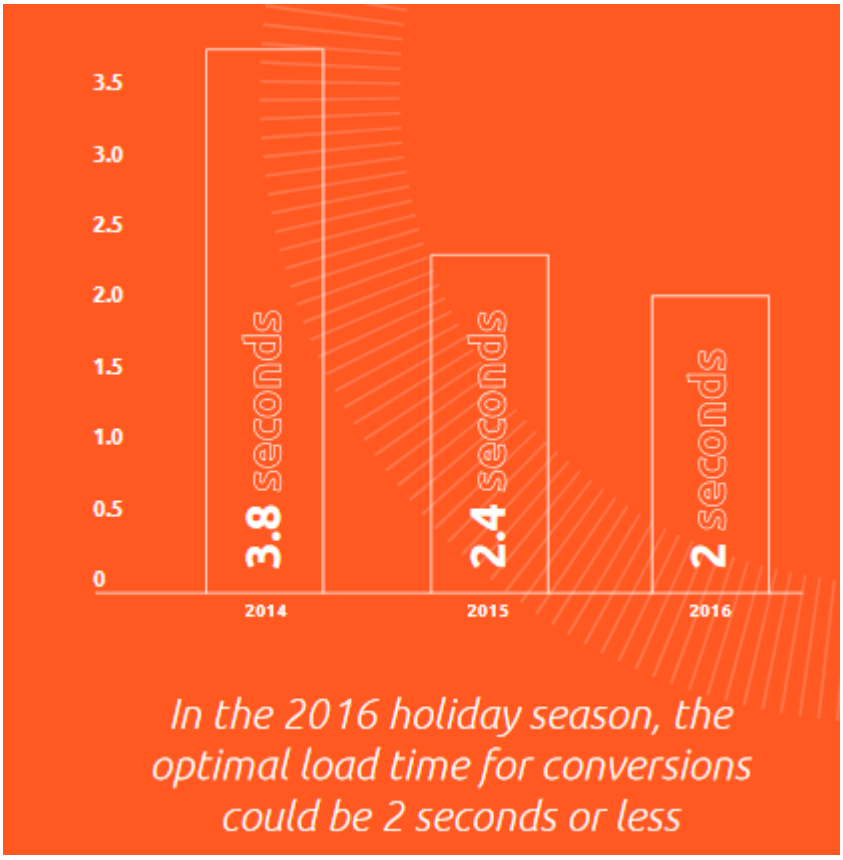
Web Forefront. (07 de Junio de 2019). *A web application's limited resources*. Obtenido de https://www.webforefront.com/performance/limitedresources.html

Wodehouse, C. (30 de Septiembre de 2016). *Django: A Python-Powered Development Framework*. Obtenido de https://www.upwork.com/hiring/development/django-programming/

ANEXOS

Anexo 1. Estadísticas acerca del tiempo de ejecución (Soasta, 2016)

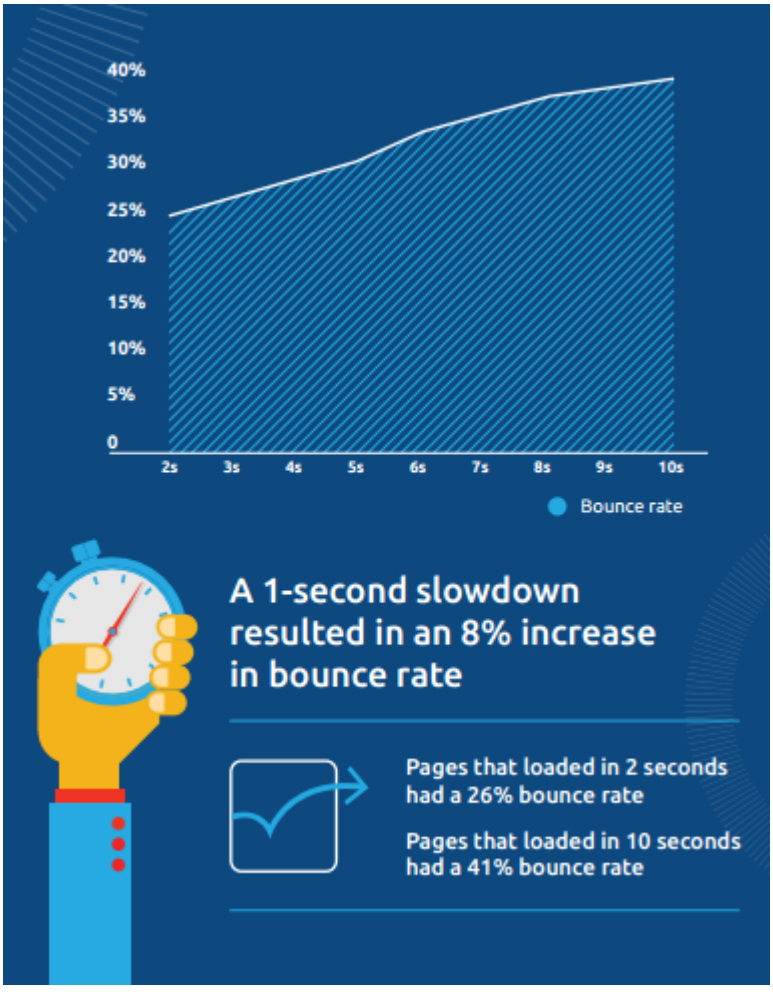
Ilustración 35. Proyección de tiempo óptimo de carga al 2016. (Soasta, 2016)



Following this trajectory, it's safe to predict that consumer expectations are likely to increase, not decrease. In the 2016 holiday shopping season, it's possible that we may see peak conversion rates correspond to load times of 2 seconds or less. (Soasta, 2016)

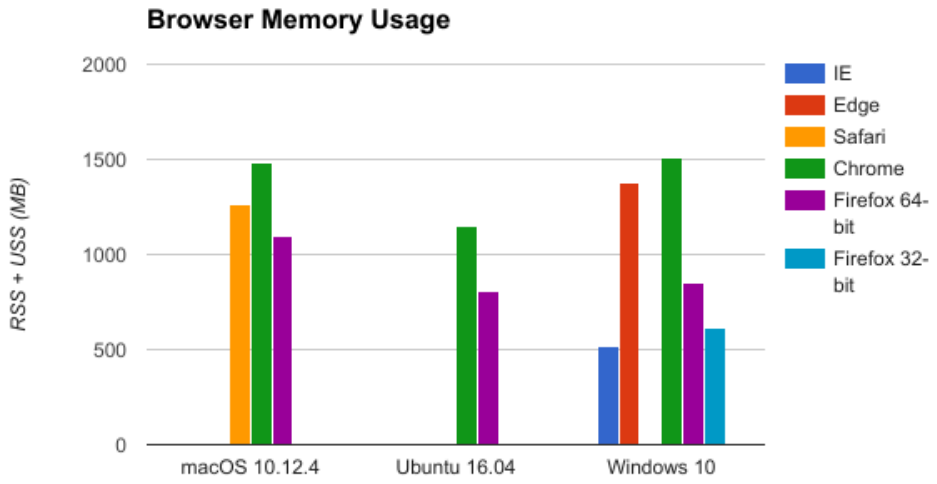
Looking at the 2015 numbers, we found a strong correlation between load time and bounce rate. For pages that loaded in 2 seconds, the median bounce rate was 26%, while pages that took 10 seconds to load experienced a 41% bounce rate. In other words, the cost of an 8-second slowdown — from 2 to 10 seconds — was a 58% increase in bounce rate.

Ilustración 36. Relación entre el tiempo de carga (load time) y la tasa de rebote (bounce rate)



Anexo 2. Estadísticas acerca del uso de memoria RAM

Ilustración 37. Relación entre el tiempo de carga (load time) y la tasa de rebote (bounce rate) (Rahm, 2017)

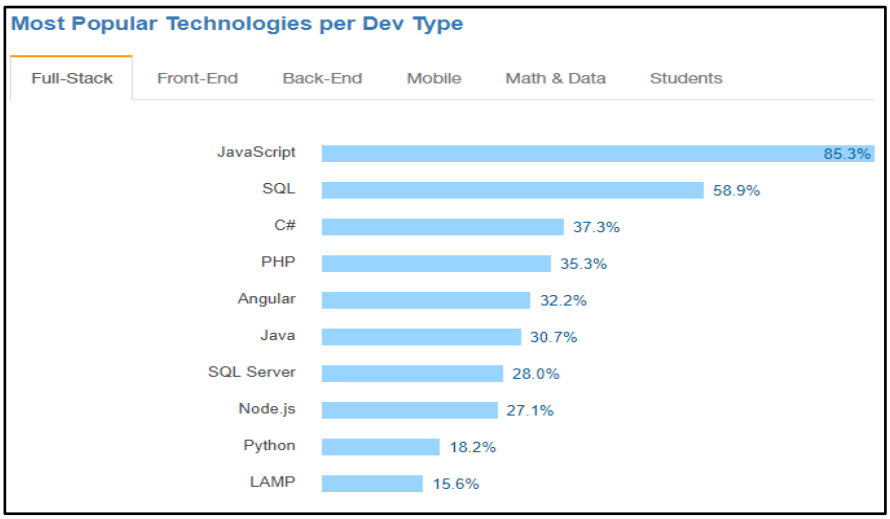


Chrome leading the pack in memory usage across the board: 2.4X the memory as Firefox 32-bit and 1.7X 64-bit on Windows. IE 11 does well, in fact it was the only one to beat Firefox. It's successor Edge, the default browser on Windows 10, appears to be striving for Chrome level consumption. On macOS 10.12 we see Safari going the Chrome route as well.

Browsers included are the default versions of IE 11 and Edge 38 on Windows, Chrome Beta 59 on all platforms, Firefox Beta 54 on all platforms, and Safari Technology Preview 29 on macOS 10.12.4.

Anexo 3. Top 10 de Tecnologías más populares

Ilustración 38. Tecnologias más populares por los programadores full-stack. Stack Overflow Developer Survey Results 2016



Anexo 4. Tendencias de Búsqueda en Google para frameworks basados en PHP y Django

La escala de variación en las búsquedas va desde 0 a 100, donde 100 representa el punto más alto en niveles de búsqueda realizadas respecto a un término o palabra clave.

Ilustración 39. Interés a lo largo del tiempo Frameworks PHP Google Trends (2017)

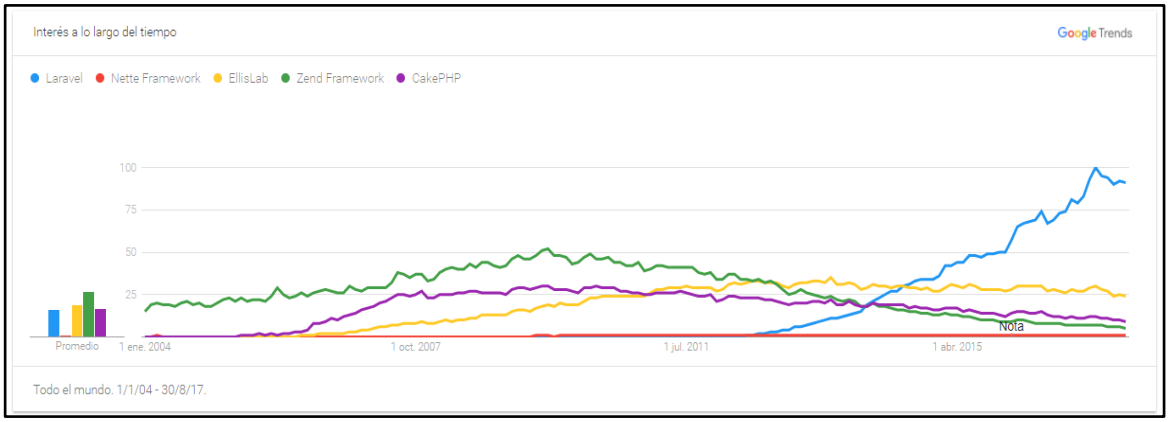


Ilustración 41. Interés a lo largo del tiempo, comparación Laravel y Django, Google Trends (2004-2017)

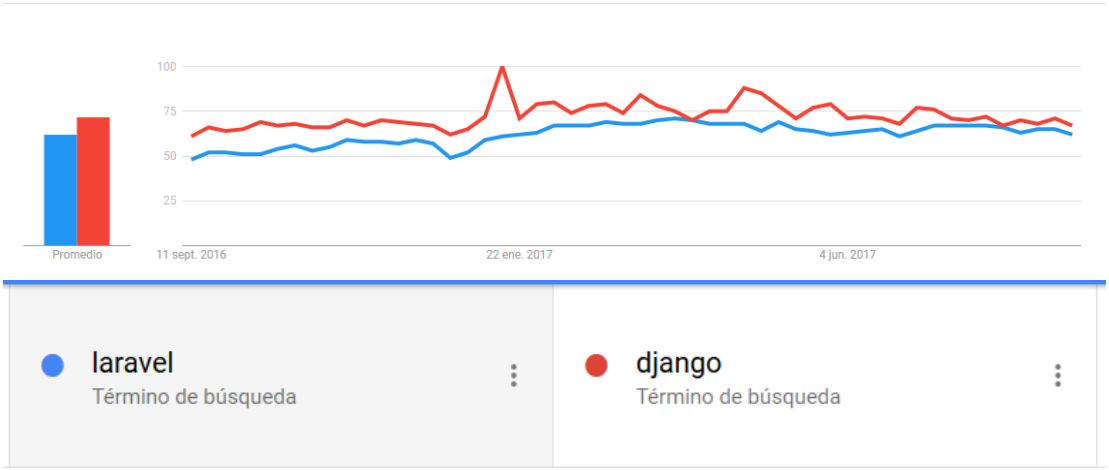
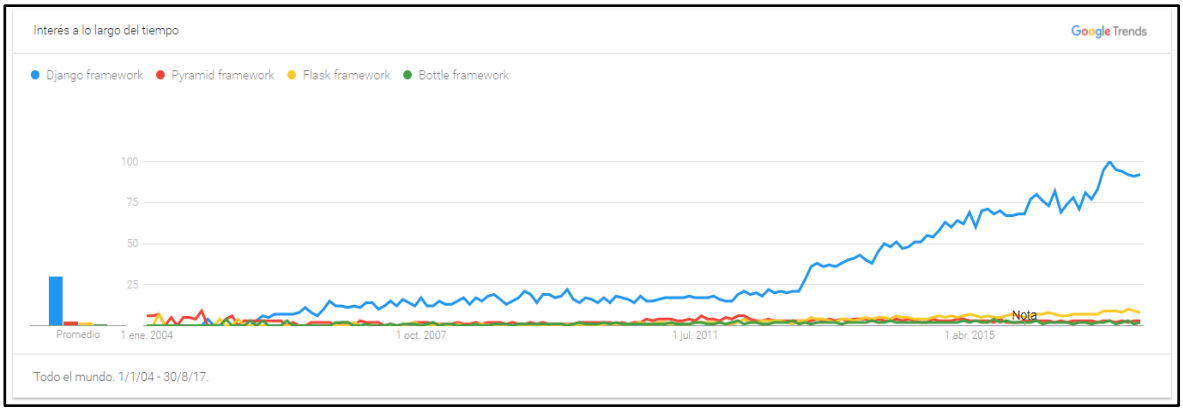
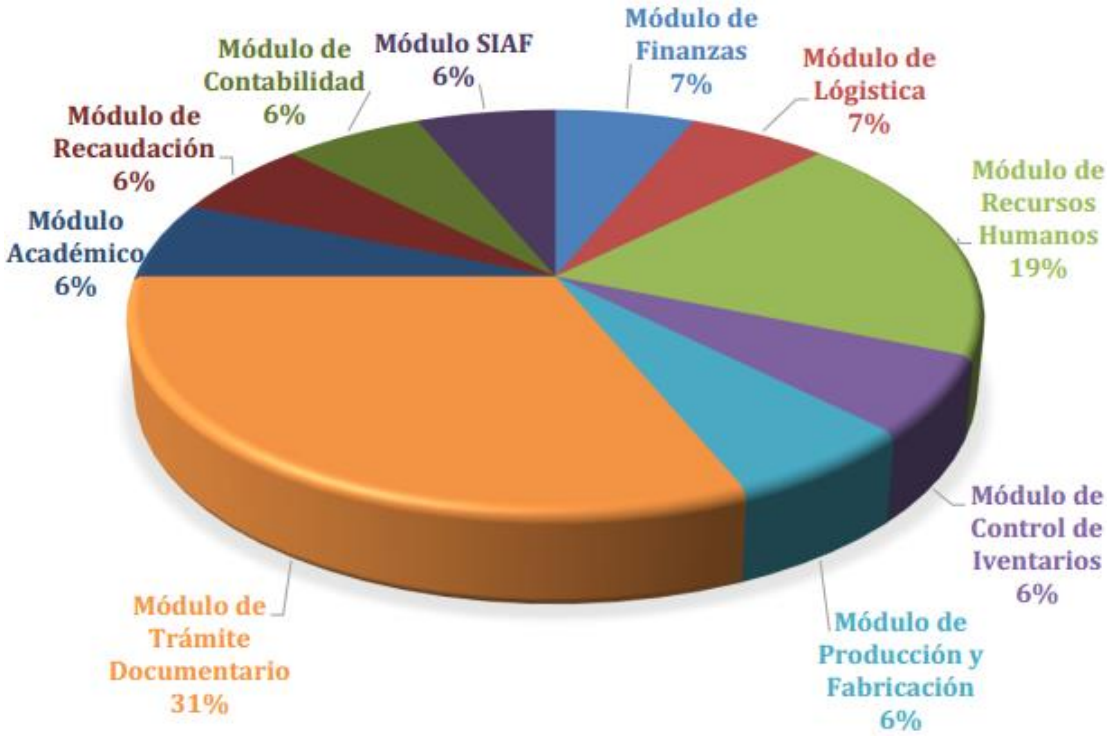


Ilustración 40. Interés a lo largo del tiempo Frameworks Python Google Trends (2017)



Anexo 5. Módulo Web más desarrollado por las empresas piuranas

Ilustración 42. Módulo más desarrollado por las empresas piuranas (Ramirez Coronado, 2017)



Los encuestados indicaron que el Módulo de Trámite Documentario (31%) es el que más se usa o desarrolla en las empresas e instituciones de Piura, seguido del Módulo de Recursos Humanos (19%). Compartiendo el tercer lugar los Módulos de Finanzas y Logística (7%) y en cuarto lugar los Módulos de Control de Inventarios, Producción y Fabricación, Académico, Recaudación y Contabilidad (6%). Cabe resaltar que estos resultados se obtuvieron debido a que las empresas e instituciones que se encuestaron son entidades públicas lo que facilitó el acceso. (Ramirez Coronado, 2017)

Anexo 6. Código fuente utilizado para la comparación de frameworks

LARAVEL

Archivo de rutas: api.php

```php
<?php

use Illuminate\Http\Request;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::group(['prefix' => 'tramite'], function() {
    Route::get('ultimoCodigo/getdata','ControllerTramite@ultimoCodigo');
    //crear tramite
    Route::post('create','ControllerTramite@store');
    //BANDEJA
    Route::get('recepcionados/getdata','ControllerMovimiento@getdataRecepcionados');
    Route::get('recibidos/getdata','ControllerMovimiento@getdataRecibidos');

    Route::get('finalizados/getdata','ControllerMovimiento@getdataFinalizados');
    Route::get('enviados/getdata','ControllerMovimiento@getdataEnviados');
    Route::get('movimientos/getdata','ControllerMovimiento@getdataMovimientos');
    Route::get('tramites/getdata','ControllerTramite@getdataTramites');

    Route::post('derivar/create','ControllerMovimiento@derivarTramite');
    Route::post('recepcionar/create','ControllerMovimiento@recepcionarTramite');
    Route::post('finalizar/create','ControllerMovimiento@finalizarTramite');

});
```

Archivo de controlador: ControllerTramite.php

Crear trámite

```php
<?php

namespace Tramite\Http\Controllers;

use Illuminate\Http\Request;
use Tramite\Movimiento;
use Tramite\Tramite;
use DB;

class ControllerTramite extends Controller
{

    public function store(Request $request){

        $rand = date('Y-m-d h:m:s');

        $request['tipo_tramite_id'] = 1;
        $request['tipo_documento_id'] = 1;
        $request['prioridad_id'] = 1;
        $request['persona_id'] = 1;
        $request['codigo'] = $rand;
        $request['folio'] = 1;
        $request['asunto'] = $rand;
        $request['unidad_destino_id'] = 1;
        $request['observacion'] = $rand;

        try {

            $success = false;

            DB::beginTransaction();

            $tramite =  Tramite::create(
                            $request->only(
                                'tipo_tramite_id',
                                'tipo_documento_id',
                                'prioridad_id',
                                'persona_id',
                                'codigo',
                                'folio',
                                'asunto'
                            )
                        );


            Movimiento::create([
                'tramite_id' => $tramite->id,
                'accion_id'  => 1,
                'estado_documento_id' => 1,
                'unidad_origen_id'  => 1,
                'unidad_destino_id' => $request->unidad_destino_id,
                'usuario_id'    => 1,
```

```
                'fecha'         => date('Y-m-d'),
                'hora'          => date('H:i:s'),
                'observacion'   => $request->observacion
            ]);

        } catch (\Exception $e)
        {
            DB::rollBack();
            $error = $e->getMessage();
            $message = "Ocurrió un error al registrar el trámite";
            return response()->json(compact('success','message','error'));
        }

        DB::rollBack();
        $success = true;
        $tramiteRegistrado = $tramite->id;
        $message = 'Trámite registrado correctamente';
        return response()-
>json(compact('success','message','tramiteRegistrado'));

    }

}
```

Archivo de controlador: ControllerMovimiento.php

Consultar bandeja de trámites finalizados

```
<?php

namespace Tramite\Http\Controllers;

use Illuminate\Http\Request;
use Tramite\Movimiento;
use Tramite\Tramite;

class ControllerMovimiento extends Controller
{
    /*
     * bandejas
    */
    public function getdataRecepcionados(){
        $recepcionados = Movimiento::where('estado',1)-
>where('estado_documento_id_id',3)->where('unidad_destino_id_id',705)-
>get();
        return response()->json($recepcionados);
    }

    public function getdataRecibidos(){
        $recibidos = Movimiento::where('estado',1)-
>where('estado_documento_id_id',4)->where('unidad_destino_id_id',705)-
>get();
        return response()->json($recibidos);
```

```
    }

    public function getdataFinalizados(){
        $finalizados = Movimiento::where('estado',1)->where('estado_documento_id',2)->where('unidad_destino_id',705)->get();
        return response()->json($finalizados);
    }

    public function getdataEnviados(){
        $enviados = Movimiento::where('estado',1)->where('estado_documento_id_id',1)->where('unidad_destino_id_id',705)->get();
        return response()->json($enviados);
    }

    public function getdataMovimientos(){
        $movimientos = Movimiento::where('estado',1)->get();
        return response()->json($movimientos);
    }

}
```

Archivo de modelo: Tramite.php

```
<?php

/**
 * Created by Reliese Model.
 * Date: Sat, 14 Jan 2017 12:29:11 -0500.
 */

namespace Tramite;

use Reliese\Database\Eloquent\Model as Eloquent;
use Tramite\Traits\TraitModel;
use Tramite\TraitsModel\TraitTramite;
/**
 * Class Tramite
 *
 */
class Tramite extends Eloquent
{
    use \Illuminate\Database\Eloquent\SoftDeletes,TraitModel,TraitTramite;
    protected $table = 'tramite_tramite';

    protected $casts = [
        'tipo_tramite_id' => 'int',
        'tipo_documento_id' => 'int',
        'prioridad_id' => 'int',
        'persona_id' => 'int',
        'folio' => 'int',
    ];

    protected $fillable = [
```

```
        'tipo_tramite_id',
        'tipo_documento_id',
        'prioridad_id',
        'persona_id',
        'codigo',
        'folio',
        'asunto',
        'url_imagen',
        'estado'
    ];

    protected $hidden  = ['created_at','updated_at','deleted_at'];

    protected static $rules = [
        'tipo_tramite_id'   => 'required|integer',
        'tipo_documento_id' => 'required|integer',
        'prioridad_id'      => 'required|integer',
        'persona_id'        => 'required|integer',
        'codigo'            => 'required|max:20',
        'folio'             => 'required|integer|min:1|max:9999',
        'asunto'            => 'required'
    ];

    protected static $messages = [
        'tipo_tramite_id.required'   => 'Seleccione el Tipo de Trámite',
        'tipo_tramite_id.integer'    => 'Seleccione el Tipo de Trámite',
        'tipo_documento_id.required' => 'Seleccione el Tipo de Documento',
        'tipo_documento_id.integer'  => 'Seleccione el Tipo de Documento',
        'prioridad_id.required'      => 'Seleccione la Prioridad',
        'prioridad_id.integer'       => 'Seleccione la Prioridad',
        'persona_id.required'        => 'Seleccione el Remitente',
        'persona_id.integer'         => 'Seleccione el Remitente',
        'codigo.required'            => 'Ingrese el número/código del tramit
e',
        'folio.required'             => 'Ingrese el folio del Trámite',
        'folio.max'                  => 'El número maximo para el folio es 9
999',
        'asunto.required'            => 'Ingrese el asunto del Trámite'
    ];

    public function persona()
    {
        return $this->belongsTo(\Usuarios\Persona::class, 'persona_id');
    }

    public function prioridad()
    {
        return $this-
>belongsTo(\Tramite\Prioridad::class, 'prioridad_id');
    }

    public function tipo_documento()
    {
        return $this-
>belongsTo(\Tramite\TipoDocumento::class, 'tipo_documento_id');
    }

    public function tipo_tramite()
    {
```

```
        return $this->belongsTo(\Tramite\TipoTramite::class, 'tipo_tramite_id');
    }

    public function movimientos()
    {
        return $this->hasMany(\Tramite\Movimiento::class, 'tramite_id');
    }

    //scopes
    public function scopePorTipoTramite($query,$idTipoTramite){
        return $query->where('tipo_tramite_id',$idTipoTramite);
    }

    public function scopePorCodigo($query,$codigo){
                return $query->where('codigo',$codigo);
            }

            public static function relaciones()
            {
                $relaciones = ["tipo_documento","prioridad","tipo_tramite","movimientos.unidad_origen",
                               "movimientos.unidad_destino","movimientos.estado_documento","persona.persona_natural"];
                return $relaciones;
            }
       }
```

Archivo de modelo: Movimiento.php

```
<?php

/**
 * Created by Reliese Model.
 * Date: Sat, 14 Jan 2017 12:29:11 -0500.
 */

namespace Tramite;

use Reliese\Database\Eloquent\Model as Eloquent;
use Tramite\Traits\TraitModel;
use Tramite\Traits\TraitBaseCrud;
use Tramite\TraitsModel\TraitMovimiento;

class Movimiento extends Eloquent
{
    use TraitModel,TraitBaseCrud,TraitMovimiento;

    protected $table = 'tramite_movimiento';
    public $timestamps = false;

    protected $casts = [
        'tipo_tramite_id' => 'int',
        'accion_id' => 'int',
```

```
25.         'estado_documento_id' => 'int',
26.         'unidad_origen_id' => 'int',
27.         'unidad_destino_id' => 'int',
28.         'usuario_id' => 'int'
29.     ];
30. 
31.     protected $dates = [
32.         //'fecha'
33.     ];
34. 
35.     protected $fillable = [
36.         'tipo_tramite_id',
37.         'accion_id',
38.         'estado_documento_id',
39.         'unidad_origen_id',
40.         'unidad_destino_id',
41.         'usuario_id',
42.         'fecha',
43.         'hora',
44.         'observacion',
45.         'estado'
46.     ];
47. 
48.     protected static $rules = [
49.         'unidad_destino_id' => 'required|integer',
50.         'observacion'       => 'required'
51.     ];
52. 
53.     protected static $messages = [
54.         'unidad_destino_id.required'=> 'Seleccione la Unidad de Destino',
55.         'unidad_destino_id.integer' => 'Seleccione la Unidad de Destino',
56.         'observacion.required'      => 'Ingrese la Observación para el Trámi
    te'
57.     ];
58. 
59.     public function accion()
60.     {
61.         return $this->belongsTo(\Tramite\Accion::class, 'accion_id');
62.     }
63. 
64.     public function estado_documento()
65.     {
66.         return $this-
    >belongsTo(\Tramite\EstadoDocumento::class, 'estado_documento_id');
67.     }
68. 
69.     public function tramite()
70.     {
71.         return $this-
    >belongsTo(\Tramite\Tramite::class, 'tipo_tramite_id');
72.     }
73. 
74.     public function user()
75.     {
76.         return $this->belongsTo(\Tramite\User::class, 'usuario_id');
77.     }
78. 
79.     public function unidad_origen()
80.     {
```

```
        return $this->belongsTo(\Tramite\Unidad::class, 'unidad_origen_id');
    }
    public function unidad_destino()
    {
        return $this->belongsTo(\Tramite\Unidad::class,'unidad_destino_id');
    }
    public static function relaciones()
    {
        $relaciones = ["unidad_origen","unidad_destino","estado_documento","accion","tramite.tipo_tramite","tramite.prioridad"];
        return $relaciones;
    }
}
```

DJANGO

Archivo de rutas: urls.py

```
from django.conf.urls import url
from .views import *
from django.contrib.auth.decorators import login_required

urlpatterns = [
############################## TRAMITE ################################
    url(r'^crear_tramite/', login_required(CrearTramite.as_view()), name='crear_tramite'),
    url(r'^update_tramite/(?P<pk>\d+)/$', login_required(TramiteUpdateView.as_view()), name='update_tramite'),
    url(r'^delete_tramite/(?P<pk>\d+)/$', login_required(TramiteDeleteView.as_view()), name='delete_tramite'),
    url(r'^listar_tramite/', login_required(TramiteListView.as_view()), name='listar_tramite'),
############################## MOVIMIENTO ################################
    url(r'^crear_movimiento/', login_required(MovimientoCreateView.as_view()), name='crear_movimiento'),
    url(r'^update_movimiento/(?P<pk>\d+)/$', login_required(MovimientoUpdateView.as_view()), name='update_movimiento'),
    url(r'^delete_movimiento/(?P<pk>\d+)/$', login_required(MovimientoDeleteView.as_view()), name='delete_movimiento'),
    url(r'^listar_movimiento/', login_required(MovimientoListView), name='listar_movimiento'),
    url(r'^enviados_search/',MovimientoEnviados, name='enviados_search'),
    url(r'^recibidos_search/',MovimientoRecibidos, name='recibidos_search'),

    url(r'^finalizados_search/',MovimientoFinalizados, name='finalizados_search'),
    url(r'^recepcionados_search/',MovimientoRecepcionados, name='recepcionados_search'),
    url(r'^enviados/',login_required(MovimientoEnvi), name='enviados'),
    url(r'^recibidos/',login_required(MovimientoReci), name='recibidos'),
```

```
    url(r'^finalizados/',login_required(MovimientoFina), name='finalizados'),
    url(r'^recepcionados/',login_required(MovimientoRecep), name='recepcionados'),
    url(r'^imagen/(?P<pk>\d+)/$',login_required(Imagen.as_view()), name='imagen'),
]
```

Archivo de vistas: views.py – Crear trámite y consultar bandeja de trámites finalizados

```
from django.shortcuts import render,redirect,render_to_response
from .models import *
from .forms import *
import re,time,json
from apps.usuarios.models import Unidad
from django.core.urlresolvers import reverse_lazy
from django.views.generic import CreateView,UpdateView,DeleteView,ListView
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from django.utils.datastructures import MultiValueDictKeyError
from django.db.models import Q
from django.views.generic.base import TemplateView
from openpyxl import Workbook
from django.http.response import HttpResponse
from openpyxl.styles import Alignment
from openpyxl.styles import Border
from openpyxl.styles import Font
from openpyxl.styles import PatternFill
from openpyxl.styles import Side

#################### TRAMITE ########################
class CrearTramite(CreateView):
    model = Tramite
    template_name = 'tramite/crear_tramite.html'
    form_class = TramiteForm
    success_url = reverse_lazy('tramite:crear_tramite')

    def post(self,request,*args,**kwargs):
        self.object = self.get_object
        form = self.form_class(request.POST or None,request.FILES or None)
        if form.is_valid():
            form.save(commit = False)
            movimiento = Movimiento(
                tipo_tramite_id = form.save(),
                accion_id = Accion.objects.get(nombre = 'Atención'),
                estado_documento_id = EstadoDocumento.objects.get(nombre = 'Enviado'),
                unidad_origen_id = request.user.unidad_id,
                unidad_destino_id = form.cleaned_data['destino'],
                usuario_id = 1,
                observacion = request.POST.get('observacion'),
                estado = '1'
            )
```

```
42.             movimiento.save()
43.             return redirect('tramite:listar_movimiento')
44.         return render(request,'tramite/crear_tramite.html',{'form':form})
45.
46.
47. ##################### MOVIMIENTO ######################
48. def MovimientoFinalizados(request):
49.     if request.user.is_superuser:
50.
51.         query = Movimiento.objects.filter(estado_documento_id = EstadoDocume
    nto.objects.values('id').filter(nombre='Finalizado'),estado = '1',unidad_ori
    gen_id = 706)
52.         lista_tipo_docu = []
53.         for movimiento in query:
54.             docu_json = {}
55.             docu_json['id'] = movimiento.id
56.             docu_json['tramite'] = movimiento.tipo_tramite_id.nombre
57.             docu_json['exp'] = movimiento.tipo_tramite_id.codigo
58.             docu_json['fecha'] = str(movimiento.fecha)
59.             docu_json['hora'] = str(movimiento.hora)
60.             docu_json['origen'] = movimiento.unidad_origen_id.nombre
61.             docu_json['accion'] = movimiento.accion_id.nombre
62.             docu_json['prioridad'] = movimiento.tipo_tramite_id.prioridad_id
    .nombres
63.             docu_json['asunto'] = movimiento.tipo_tramite_id.asunto
64.             docu_json['estado'] = movimiento.estado_documento_id.nombre

65.             docu_json['tramite_id'] = movimiento.tipo_tramite_id
66.             lista_tipo_docu.append(docu_json)
67.         data = json.dumps(lista_tipo_docu)
68.
69.         return HttpResponse(data, 'application/json')
70.     else:
71.         query = Movimiento.objects.filter(estado_documento_id = EstadoDocume
    nto.objects.values('id').filter(nombre='Finalizado'),estado = '1',unidad_ori
    gen_id = 706)
72.
73.         lista_tipo_docu =[]
74.         for movimiento in query:
75.             docu_json = {}
76.             docu_json['id'] = movimiento.id
77.             docu_json['tramite'] = movimiento.tipo_tramite_id.tipo_tramite_i
    d.nombre
78.             docu_json['exp'] = movimiento.tipo_tramite_id.codigo
79.             docu_json['fecha'] = str(movimiento.fecha)
80.             docu_json['hora'] = str(movimiento.hora)
81.             docu_json['origen'] = movimiento.unidad_origen_id.nombre
82.             docu_json['accion'] = movimiento.accion_id.nombre
83.             docu_json['prioridad'] = movimiento.tipo_tramite_id.prioridad_id
    .nombre
84.             docu_json['asunto'] = movimiento.tipo_tramite_id.asunto
85.             docu_json['estado'] = movimiento.estado_documento_id.nombre

86.             docu_json['tramite_id'] = movimiento.tipo_tramite_id.id
87.             lista_tipo_docu.append(docu_json)
88.         data = json.dumps(lista_tipo_docu)
89.         return HttpResponse(data, 'application/json')
90.     return render(request,'tramite/movimiento/finalizados.html')
```

Archivo de modelos: models.py

```
1.  # -*- coding: utf-8 -*-
2.  from django.db import models
3.  from .validators import validate_even
4.  from apps.usuarios.models import Unidad,Persona,User,PersonaNatural
5.
6.  class Tramite(models.Model):
7.      """Model definition for Tramite."""
8.      # TODO: Define fields here
9.      id = models.AutoField(primary_key = True)
10.     tipo_tramite_id = models.ForeignKey(TipoTramite,blank = True, null = Tru
    e, db_column='tipo_tramite_id')
11.     tipo_documento_id = models.ForeignKey(TipoDocumento,blank = True, null =
    True, db_column='tipo_documento_id')
12.     prioridad_id = models.ForeignKey(Prioridad, db_column='prioridad_id')
13.     persona_id = models.ForeignKey(Persona, db_column='persona_id')
14.     destino = models.ForeignKey(Unidad, on_delete = models.CASCADE,blank = T
    rue, null = True, db_column='destino_id')
15.     codigo = models.CharField('Nombre de Tramite', max_length=200,unique = T
    rue)
16.     dias_atencion = models.IntegerField(blank = True, null = True)
17.     folio = models.CharField('Folio', max_length=255)
18.     asunto = models.CharField('Asunto',max_length = 255)
19.     url_image = models.ImageField('Imagen', upload_to=upload_location,
20.         null=True,blank=True,
21.         height_field = "height_field",
22.         width_field = "width_field"
23.         )
24.     height_field = models.IntegerField(default = 400)
25.     width_field = models.IntegerField(default = 400)
26.     deleted_at = models.TimeField('Deleted', auto_now = False, auto_now_add
    = True)
27.     created_at = models.TimeField('Created', auto_now = False, auto_now_add
    = True)
28.     updated_at = models.TimeField('Updated', auto_now = False, auto_now_add
    = True)
29.
30.     class Meta:
31.         """Meta definition for Tramite."""
32.
33.         verbose_name = 'Tramite'
34.         verbose_name_plural = 'Tramites'
35.
36.     def __str__(self):
37.         """Unicode representation of Tramite."""
38.         return self.codigo
39.
40.
41. class Movimiento(models.Model):
42.     """Model definition for Movimiento."""
43.     # TODO: Define fields here
44.     id = models.AutoField(primary_key = True)
45.     tipo_tramite_id = models.ForeignKey(Tramite,blank = True, null = True, d
    b_column='tipo_tramite_id')
46.     accion_id = models.ForeignKey(Accion,blank = True, null = True,default =
    1, db_column='accion_id')
```

```
47.     estado_documento_id = models.ForeignKey(EstadoDocumento,blank = True, nu
    ll = True,default = 1, db_column='estado_documento_id')
48.     unidad_origen_id = models.ForeignKey(Unidad,blank = True, null = True,re
    lated_name='origen', db_column='unidad_origen_id')
49.     unidad_destino_id = models.ForeignKey(Unidad,blank = True, null = True,r
    elated_name='destino', db_column='unidad_destino_id')
50.     usuario_id = models.ForeignKey(Persona,blank = True, null = True, db_col
    umn='usuario_id')
51.     fecha = models.DateField('Fecha', auto_now=False, auto_now_add=True)
52.     hora = models.TimeField('Hora', auto_now=False, auto_now_add=True)
53.     observacion = models.TextField('Observación',blank = True, null = True)

54.     estado = models.CharField('Estado',max_length = 1)
55.     deleted_at = models.TimeField('Deleted', auto_now = False, auto_now_add
    = True)
56.     created_at = models.TimeField('Created', auto_now = False, auto_now_add
    = True)
57.     updated_at = models.TimeField('Updated', auto_now = False, auto_now_add
    = True)
58.
59.     class Meta:
60.         """Meta definition for Movimiento."""
61.
62.         verbose_name = 'Movimiento'
63.         verbose_name_plural = 'Movimientos'
64.
65.     def __str__(self):
66.         """Unicode representation of Movimiento."""
67.         return str(self.id)
```

Anexo 7. Características de equipos descartados para hacer pruebas

Equipo #1

Componente	Descripción
Procesador	Intel Celeron ® CPU N2830 2.16GHz
Disco duro	1TB
Memoria RAM	4.00 GB (3.89 utilizable)
Sistema Operativo	Windows 10 Single Language 64 bits

Equipo #2

Componente	Descripción
Procesador	Intel Core i3 - 6006u 2.0GHz
Disco duro	1TB
Memoria RAM	4.00 GB (3.89 utilizable)
Sistema Operativo	Windows 10 Home 64 bits